Caging and Path Non-Existence: a Deterministic Sampling-Based Verification Algorithm

Anastasiia Varava, J. Frederico Carvalho, Florian T. Pokorny, and Danica Kragic

Abstract Caging restricts the mobility of an object without necessarily immobilizing it completely. The object is caged if it cannot move arbitrarily far from its initial position. Apart from its common applications to grasping and manipulation, caging can also be considered as a problem dual to motion planning: an object is caged when it is isolated within a bounded connected component of its configuration space and is disconnected from the rest of the latter. In this paper, we address the problem of caging and path non-existence verification in 2D and 3D workspaces by representing a subset of the collision space as a simplicial complex and analyzing the connectivity of its complement. Since configuration spaces of 2D and 3D rigid objects are three-dimensional and six-dimensional respectively, it is computationally expensive to reconstruct them explicitly. Thus, we represent the object's collision space as a union of a finite set of 'slices', corresponding to small intervals of the object's orientation coordinates. We also discuss possible generalizations of our approach to higher dimensions.

1 Introduction

A *cage* is an arrangement of obstacles that restricts the mobility of the object in such a way that it cannot escape arbitrarily far from its initial position. In robotics, caging relates to grasping, where it is used either as an alternative to contact-based grasping or as a pregrasping condition [8, 9, 13, 16, 17, 20]. Apart from that, caging has applications in multi-agent manipulation [18].

In terms of the object's configuration space, the object is caged if its configuration lies in a bounded connected component of its free space. Therefore, a cage and a path can be seen as two opposite concepts: while the existence of a path ensures

A. Varava, J. F. Carvalho, F. T. Pokorny, and D. Kragic
KTH Royal Institute of Technology, Stockholm, Sweden
e-mail: {varava, jfpbdc, fpokorny, dani}@kth.se

reachability, the presence of a cage means that the object's mobility is restricted to the current bounded connected component, see Fig. 1.

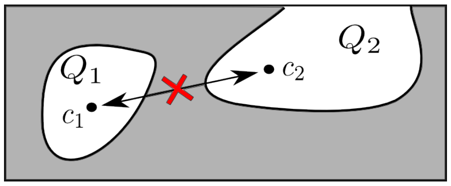

Fig. 1 The collision space of the object is illustrated in grey, the two connect components of the free space are denoted by Q_1 and Q_2. We see that the configuration c_1 is caged as Q_1 is bounded. Having Q_1 and Q_2 we can also say that any configuration $c_2 \in Q_2$ cannot be reached from c_1.

The problem of proving path non-existence can be applied to path planning. Most of the sampling-based planning algorithms perform iterative random sampling and incrementally build paths between configurations. Usually these algorithms are either probabilistically complete [1] or resolution complete [6], but they are not guaranteed to find a path in finite time if one exists. Therefore, one of the practical questions is how long the planner should search for a path before concluding that it does not exists. Instead of using stopping heuristics, one could avoid this problem by proving path non-existence. The latter is a challenging problem as it requires retrieving information about the connectivity of the entire configuration space rather than finding a single path which has been addressed in [2, 10, 22].

Essentially, computing the connected components of the free space would solve two problems: *(i)* proving or disproving caging and *(ii)* checking if the object can be moved from one configuration to another. In this paper, we address both problems for rigid objects in 2D and 3D workspaces.

2 Related Work

The notion of a planar cage was initially introduced by Kuperberg in 1990 [5] as a set of n coplanar points lying in the complement of a polygon and preventing it from escaping arbitrarily far from its initial position. This problem can be directly applied to robotics if we represent the fingertips or mobile robots as points, and the object as a polygon. Point-based caging of polygons and polytopes has received a lot of attention in the context of grasping and multi-agent manipulation. Rimon and Blake [14] proposed an algorithm computing a set of caging configurations of a two-fingered hand for planar non-convex objects. In [11], Pipattanasomporn and Sudsang proposed an algorithm reporting all two-finger caging sets for a given concave polygon. Vahedi and van der Stappen extended this result in [21] by proposing an algorithm that returns all caging placements of a third finger given a polygonal object and a placement of two other fingers. A similar approach has also been adopted for caging 3D objects. Pipattanasomporn and Sudsang [12] proposed an algorithm for computing all two-finger cages for non-convex polytopes. Rodriguez et al. introduced a notion of a pregrasping cage in their work [15]. Starting from a pregrasping cage, a manipulator can move to a form closure grasp without breaking the cage.

In the above mentioned works fingertips are represented as points or spheres, and more complex shapes of caging tools usually are not taken into account. Alternatively, one can derive sufficient caging conditions based on specific geometric and topological features of the object and the caging tools. In particular, Stork et al. proposed an approach towards caging objects with 'holes' in their works [13, 16, 17]. Subsequently, Varava et al. considered loop-based caging of objects with narrow parts – so-called 'necks' and 'double forks' [20]. Sufficient conditions for caging objects of particular shapes have also been studied in [8, 9].

The advantage of the above mentioned approaches is that they utilize global geometric properties of the objects. Therefore, one does not have to explicitly reconstruct the free space of the object, which is a challenging task due to its high dimensionality. However, these methods require the objects to exhibit particular shape features, which prevents us from applying them to caging objects of arbitrary shape. A more general approach is to directly analyze the topological properties of the configuration space of the object.

In this context, caging is closely related to the problem of proving path non-existence. Motivated by motion planning applications, in their pioneering work [2] Basch et al. propose an approach towards proving that two configurations are disconnected because the object is 'too big' or 'too long' to pass through a 'gate' between them. The authors apply their algorithm to several simple objects and integrate it with a PRM motion planner. In [22], Zhang et al. use approximate cell decomposition and prove path non-existence. They decompose a configuration space into a set of cells and for each cell decide if it lies in the collision space. In [10] McCarthy et al. propose a somewhat similar approach. There, they randomly sample the configuration space and reconstruct its approximation as an alpha complex. They later use it to check the connectivity between pairs of configurations.

In this paper, we also aim to study the connectivity of the free space of the object. However, unlike [10], we do not construct the collision space directly. Instead we decompose it into a finite set of lower dimensional 'slices'. We compute the connected components of the latter and analyze the possible transitions between different slices. This allows us to overcome the dimensionality problem without losing any necessary information about the topology of the configuration space.

3 An Overview of Our Method

In this section we state the problem we are going to address, introduce the necessary notation, and provide an intuitive description of our approach while avoiding going into technical details.

3.1 Problem Formulation

Let us start with the necessary definitions. A *rigid object* $\mathscr{O}$ is a compact connected subset of $\mathbb{R}^d$ of the form $\mathscr{O} = cl(U)$, where $d \in \{2,3\}$, U is an open subset of $\mathbb{R}^d$, and $cl(U)$ denotes the closure of U. A *set of obstacles* $\mathscr{S}$ is a compact subset of $\mathbb{R}^d$ with the same property: it can be represented as a closure of an open subset of $\mathbb{R}^d$. Let $\mathscr{C} = SE(d)$ denote the configuration space of the object. We define its collision space as the set of the objects configurations in which the object penetrates the obstacles: $\mathscr{C}^{col} = \{c \in \mathscr{C} \,|\, \mathrm{int}(c(\mathscr{O})) \cap \mathrm{int}(\mathscr{S}) \neq \emptyset\}$, where $c(\mathscr{O})$ denotes the object in a configuration c. Note that this definition allows the object to be in contact with the obstacles. The free space $\mathscr{C}^{free}$ is the complement of the collision space: $\mathscr{C}^{free} = \mathscr{C} - \mathscr{C}^{col}$. We assume that both the obstacles and the object can be approximated as unions of balls lying in their interior, $\mathscr{S} = \{B_{R_1}(X_1), .., B_{R_n}(X_n)\}$ and $\mathscr{O} = \{O_{r_1}(Y_1), .., O_{r_m}(Y_m)\}$ of radii $R_1, .., R_n$ and $r_1, .., r_m$ respectively[1].

Our goal is to provide an algorithm which:

- takes the approximations of the object and the obstacles as input;
- constructs an approximation of the collision space;
- for a given configuration, checks whether it is caged; returns “True” if it is guaranteed to be caged, and “Undefined” otherwise;
- for a given pair of configurations, checks whether there is a collision-free path between them; returns “False” if they are guaranteed to be disconnected, and “Undefined” otherwise.

3.2 Our Approach: The General Idea

To address our problem we compute an approximation of $\mathscr{C}^{col}$, and then analyse the connectivity of its complement – the free space. Let us think of $\mathscr{C}$ as a product $\mathscr{C} = \mathbb{R}^d \times SO(d)$ where $d \in \{2,3\}$.

Note that as long as the object is caged by a subset of the collision space, it is guaranteed to be caged by the entire collision space. Therefore, to solve our problem, it is not necessary to reconstruct the exact collision space $\mathscr{C}^{col}$. Instead, it is enough to approximate it by its sufficiently large subset $\mathscr{C}^{col}_a \subset \mathscr{C}^{col}$.

Instead of reconstructing $\mathscr{C}^{col}$ directly, we work with its projections $\mathscr{C}^{col}(\mathscr{O}^{\phi})$ onto $\mathbb{R}^d$ corresponding to fixed orientation ϕ of the object in $SO(d)$.

Since the number of all possible orientation values is infinite, we divide $\mathscr{C}$ into a finite number of slices:

Definition 1 *A slice of the configuration space $\mathscr{C}$ is a subset of $\mathscr{C}$ defined as follows:*

$$Sl_U = \mathbb{R}^d \times U,$$

[1] From now on, we will always use this approximation in this paper, and hence we write the approximations as equalities to simplify the notation.

where U is an open subset of $SO(d)$.

Similarly, we denote a slice of the collision (free) space by Sl_U^{col} (Sl_U^{free}). For each slice we build an approximation of the collision space Sl_U^{col}, so that the union of all of them represents the entire collision space:

$$\mathscr{C}^{col} = \bigcup_{U \subset SO(d)} Sl_U^{col}$$

Let us now discuss how we construct the slices. Note that to show that the object collides with an obstacle it is enough to show that some part of the object is in collision. We define an $\varepsilon-$core of the object as follows:

Definition 2 *An ε-core of an object $\mathscr{O}$ is a subset $\mathscr{O}_\varepsilon$ of the object such that any point of $\mathscr{O}_\varepsilon$ lies at least at a distance*[2] *ε from the boundary of $\mathscr{O}$:*

$$\mathscr{O}_\varepsilon = \{p \in \mathscr{O} | d(p, \partial\mathscr{O}) \geq \varepsilon\}$$

Let $\mathscr{O}^\phi$ and $\mathscr{O}_\varepsilon^\phi$ denote an object $\mathscr{O}$ and its ε-core with a fixed orientation $\phi \in SO(d)$, respectively. Note that the collision space of an object with a fixed rotation is a subset of $\mathbb{R}^d$.

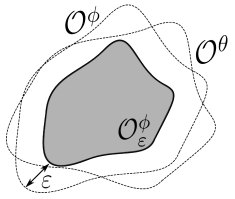

Fig. 2 An ε-core $\mathscr{O}_\varepsilon^\phi$ of the object $\mathscr{O}^\phi$ is depicted in grey; it is contained within the rotated object $\mathscr{O}_\varepsilon^\theta$.

In Sec 4, we show that for an object $\mathscr{O}^\phi$ and its ε-core $\mathscr{O}_\varepsilon^\phi$ with a fixed orientation ϕ we can find an open neighbourhood $U(\varepsilon,\phi) \subset SO(d)$ of ϕ such that for any $\theta \in U(\varepsilon,\phi)$ the ε-core $\mathscr{O}_\varepsilon^\phi$ is fully contained within a slightly rotated object $\mathscr{O}^\theta$, see Fig. 2. Since $SO(d)$ is compact, we choose a finite set of rotations $\{\phi_1,..,\phi_s\}$ so that the corresponding family of sets $U(\varepsilon,\phi_i)$ is a finite open cover of $SO(d)$: $SO(d) = \bigcup_{i\in\{1,..,s\}} U(\varepsilon,\phi_i)$.

Now note that if $\mathscr{O}_\varepsilon^\phi \subset \mathscr{O}^\theta$, then $\mathscr{C}^{col}(\mathscr{O}_\varepsilon^\phi) \subset \mathscr{C}^{col}(\mathscr{O}^\theta)$. Thus, we can approximate the collision space of a slice by $aSl_{U(\varepsilon,\phi)}^{col}$ defined as follows:

$$Sl_{U(\varepsilon,\phi)}^{col} \approx aSl_{U(\varepsilon,\phi)}^{col} = \mathscr{C}^{col}(\mathscr{O}_\varepsilon^\phi) \times U(\varepsilon,\phi),$$

and then $\mathscr{C}^{col} \approx \mathscr{C}_a^{col} = \bigcup_{i\in\{1,..,s\}} aSl_{U(\varepsilon,\phi_i)}^{col}$.

[2] By distance here we mean Euclidean distance in $\mathbb{R}^d$

Given an object $\mathscr{O}^{\phi}$ and an $\varepsilon > 0$, we compute the corresponding ε-core $\mathscr{O}^{\phi}_{\varepsilon}$. We then compute an approximation of the collision space for $\mathscr{O}^{\phi}_{\varepsilon}$. Recall that the set of obstacles is represented as a set of balls $S = \{B_{R_1}(X_1), .., B_{R_n}(X_n)\}$ and $\mathscr{O} = \{O_{r_1}(Y_1), .., O_{r_m}(Y_m)\}$. The ε-core can be represented as a set of balls with centres at the same points and smaller radii: $\mathscr{O}^{\phi}_{\varepsilon} = \{O_{r_1-\varepsilon}(Y_1), .., O_{r_m-\varepsilon}(Y_m)\}$.

However, to understand the connectivity of the free space it is not enough to consider different orientation intervals separately. Consider an example, see Fig.3. Here we have two slices of the free space corresponding to two orientation intervals $U_1 \cap U_2 \neq \emptyset$. Both slices have 3 connected components: Q_1, Q_2, Q_3, and Q'_1, Q'_2, Q'_3 respectively. The object can move between the connected components Q_i and $Q'_j$from different slices if *(i)* the corresponding orientation intervals have a non-empty intersection and *(ii)* Q_i and Q_j have non-empty intersection in $\mathbb{R}^d$.

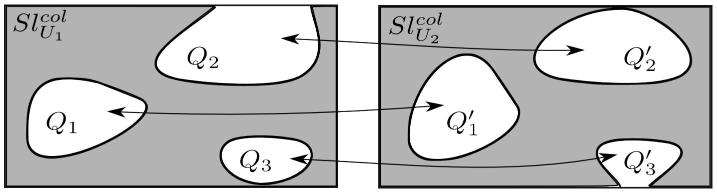

Fig. 3 Two approximations of the collision space corresponding to the non-disjoint orientation intervals U_1 and U_2 are depicted in grey. The free space is depicted in white. The arrows between different connected components indicate the possible transitions between slices.

Therefore, the object is contained within a bounded connected component of the free space if *(i)* for the corresponding slice Sl_U of the configuration space it is contained within a bounded connected component of $\mathscr{C}^{free}$, and *(ii)* there is no path between slices leading the object to an unbounded connected component of $\mathscr{C}^{free}$.

4 Methodology

In this section, we describe the technical details of our approach. We start with the derivation of the possible slice size depending on the size of the chosen ε-core of the object.

4.1 The Object Core and The Size of The Slices

Consider an ε-core of the object in a given orientation for a fixed ε, and let us fix its orientation $\phi \in SO(d)$. We now want to explicitly find an open set $U(\varepsilon, \phi) \subset SO(d)$ such that for any $\theta \in U(\varepsilon, \phi)$ the ε-core $\mathscr{O}^{\phi}_{\varepsilon}$ is fully contained within any $\mathscr{O}^{\theta}$. We represent a motion in $SO(d)$ as a rotation matrix R. First, we make the following observation:

Observation 1 *Consider an object $\mathcal{O}^{\phi}$ and an ε-core $\mathcal{O}^{\phi}_{\varepsilon}$ inside it. Let us fix the orientation $\phi \in SO(d)$ of $\mathcal{O}^{\phi}_{\varepsilon}$, and apply a rotation R to $\mathcal{O}^{\phi}$. Then $\mathcal{O}^{\phi}_{\varepsilon}$ is fully contained inside the rotated object $\mathcal{O}^{R\phi}$ if the displacement $||Ry-y||_2$ of a point y after applying the rotation is smaller than ε for any point $y \in \mathcal{O}^{\phi}$.*

The proof is trivial, as by construction any point of $\mathcal{O}^{\phi}_{\varepsilon}$ is separated at least at a distance ε from the boundary of $\mathcal{O}^{\phi}$. Thus, if $\mathcal{O}^{\phi}_{\varepsilon}$ is not lying inside of $\mathcal{O}^{R\phi}$, then there exists a point $y \in \mathcal{O}^{\phi}$ such that the distance from it to its image Ry in $\mathcal{O}^{R\phi}$ is not smaller than ε, $||Ry-y||_2 \geq \varepsilon$.

Note that in fact the form of $U(\varepsilon,\phi)$ does not depend on ϕ, and $U(\varepsilon,R\phi)$ can be obtained by rotating the former: $U(\varepsilon,R\phi) = RU(\varepsilon,\phi)$ for any $\phi \in SO(d)$ and any rotation R.

We now want to estimate the maximum displacement $||Ry-y||_2$ of a point $y \in \mathbb{R}^d$ after applying a rotation R. Let I be the identity matrix. We call $R-I$ a displacement matrix associated to the rotation R. To estimate $||(R-I)y||_2$, we compute the spectral norm of the displacement matrix:

$$||R-I||_2 = \sup_{y\in\mathbb{R}^3,\ ||y||_2\neq 0} \frac{||(R-I)y||_2}{||y||_2},$$

which gives us an upper bound for the displacement:

$$\forall y \in \mathcal{O}^{\phi}:\ ||R-I||_2 \max_{x\in\mathcal{O}^{\phi}} (||x||_2) \geq ||(R-I)y||_2$$

4.1.1 2D Workspace

Consider a two-dimensional workspace. In this case, the configuration space (and hence the collision space) of the object is a three-dimensional subset of $SE(2) = \mathbb{R}^2 \times SO(2)$, where each configuration is described by a position of the object $x \in \mathbb{R}^2$ and its orientation $\phi \in SO(2)$.

In this case rotation matrix can be written as

$$R_\theta = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix},$$

Recall that the spectral norm of $R-I$ is defined as the square root of the maximum eigenvalue of $(R-I)^T(R-I)$.

This matrix has a single single eigenvalue of algebraic multiplicity two:

$$\lambda = (\cos(\theta)-1)^2 + \sin^2(\theta) = 2-2\cos(\theta) = 4\sin^2(\theta/2),$$

which gives us an upper bound for the maximum displacement of y after applying a rotation R_θ:

$$||R_\theta y - y||_2 \leq 2|\sin(\theta/2)| \cdot ||y||_2$$

4.1.2 3D Workspace

We represent motions in $SO(3)$ as a composition of the pitch, yaw and roll rotation matrices:

$$R_\phi = \begin{bmatrix} \cos(\phi) & 0 & \sin(\phi) \\ 0 & 1 & 0 \\ -\sin(\phi) & 0 & \cos(\phi) \end{bmatrix},\ R_\sigma = \begin{bmatrix} \cos(\sigma) & -\sin(\sigma) & 0 \\ \sin(\sigma) & \cos(\sigma) & 0 \\ 0 & 0 & 1 \end{bmatrix},\ R_\theta = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}$$

As before, we want to estimate the displacement incurred by applying a given rotation. Consider a fixed triplet (θ, σ, ϕ) we define $R_{(\theta,\sigma,\phi)} = R_\theta \cdot R_\sigma \cdot R_\phi$, and once again compute the norm of the displacement matrix as the square root of the maximum eigenvalue of $(R_{(\theta,\sigma,\phi)} - I)^T(R_{(\theta,\sigma,\phi)} - I)$.

Direct computations show that this matrix has a single non-zero eigenvalue of algebraic multiplicity two:

$$\lambda = 3 - \cos(\sigma)\cos(\phi) - \cos(\theta)\cos(\phi) - \cos(\sigma)\cos(\theta) - \sin(\sigma)\sin(\phi)\sin(\theta)$$

We can simplify this expression by rewriting the products of trigonometric functions as their sums and get:

$$\lambda = 3 - \frac{\cos(\sigma-\phi)}{2} - \frac{\cos(\sigma+\phi)}{2} - \frac{\cos(\sigma-\theta)}{2} - \frac{\cos(\sigma+\theta)}{2} - \frac{\cos(\theta-\phi)}{2} \\ - \frac{\cos(\theta+\phi)}{2} - \frac{\sin(\sigma+\theta-\phi)}{4} - \frac{\sin(\sigma-\theta+\phi)}{4} + \frac{\sin(\sigma+\theta+\phi)}{4} + \frac{\sin(\sigma-\theta-\phi)}{4}$$

For simplicity, instead of treating our parameters θ, σ, and ϕ separately, we consider those triples which lie inside of a ball $B_r(0)$ of radius $r > 0$.

Now for a fixed $r > 0$ we observe that for any $(\theta, \sigma, \phi) \in B_r(0)$, each of the arguments $\sigma \pm \theta, \sigma \pm \phi, \theta \pm \phi, \sigma \pm \phi \pm \theta$ is contained in $[-r, r]$. Indeed, for example

$$|s + t + p| \leq |s| + |t| + |p| \leq \sqrt{s^2 + t^2 + p^2} \leq r,$$

and the same follows for any other argument from the list. This means that for any r in the interval $[-\pi/2, \pi/2]$ we have:

$$\cos(\sigma+\theta), \cos(\sigma-\theta), \cos(\sigma+p), \cos(\sigma-\phi), \cos(\theta+\phi), \cos(\theta-\phi) \geq \cos(r)$$
$$\sin(\sigma+\phi+\theta), \sin(\sigma-\theta-\phi) \geq \sin(-r) = -\sin(r)$$
$$\sin(\sigma+\theta-\phi), \sin(\sigma-\theta+\phi) \geq -\sin(r),$$

And hence

$$\lambda \leq 3 - 3\cos(r) + \sin(r)$$

Furthermore, since this is true for every $l \in [0,r]$ we get

$$||R_{(\theta,\sigma,\phi)} - I||_2 \leq \sqrt{3-3\cos(r)+\sin(r)} \qquad \forall(\theta,\sigma,\phi) \in B_r(0),$$

and so the maximum displacement of y after applying a rotation $R_{(\theta,\sigma,\phi)}$ can be estimated as follows:

$$||(R_{(\theta,\sigma,\phi)} - I)y||_2 \leq \sqrt{3-3\cos(r)+\sin(r)} \cdot ||y||_2$$

Fig. 4 illustrates how the displacement grows with the radius r.

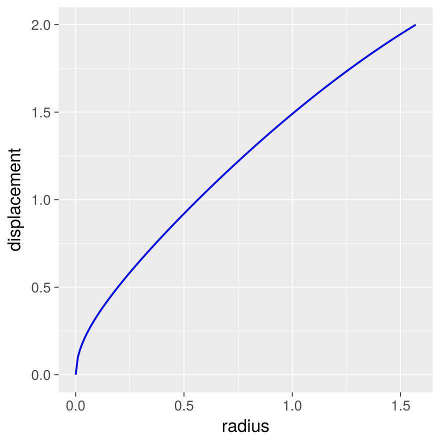

Fig. 4 Graph of the maximum displacement estimate for a ball of radius r

4.2 Collision Space Reconstruction

Let us now discuss how we construct the collision space for each slice. Recall that both the object and the obstacles are represented as unions of balls, $S = \bigcup_{i\in\{1,..,n\}} B_{R_i}(X_i)$ and $\mathscr{O} = \bigcup_{i\in\{1,..,m\}} O_{r_i}(Y_i)$ of radii $R_1,..,R_n$ and $r_1,..,r_m$ respectively. Let the frame of the object be centred in its centroid G, $\overline{GY_i}$ denote the position vector of the center of each ball, $i \in \{1..m\}$, and $\mathrm{T}_{-\overline{GY_i}}$ denote a translation to a vector $-\overline{GY_i}$.

The object represented as a union of balls collides with the obstacles if at least one of the balls is in collision, so the collision space of the object is a union of the collision spaces of the balls shifted with respect to the position of the balls centers:

$$\mathscr{C}^{col}(\mathscr{O}^{\phi}_{\varepsilon}) = \bigcup_{i\in\{1..m\}} \mathrm{T}_{-\overline{GY_i}}(\mathscr{C}^{col}(O_{r_i-\varepsilon}))$$

Note also that a ball collides with the obstacles when the distance from the obstacles to its center is not greater than the radius of the ball, so the collision space of a single ball of radius $r_i - \varepsilon$ can be expressed as follows:

$$\mathscr{C}^{col}(O_{r_i-\varepsilon}) = \{x \in \mathbb{R}^d | \, \mathrm{d}(x,S) \leq r_i - \varepsilon\} = \bigcup_{j\in\{1..n\}} B_{R_j+r_i-\varepsilon}(X_j)$$

So, the collision space of the ε-core can be written as

$$\mathscr{C}^{col}(\mathscr{O}_\varepsilon^\phi) = \bigcup_{i\in\{1..m\},\ j\in\{1..n\}} \mathrm{T}_{-\overline{GY_i}}(B_{R_j+r_i-\varepsilon}(X_j))$$

Algorithm 1: Construct-Slice

```
Data: An ε-core $\mathscr{O}_\varepsilon^\phi = \{O_{r_1-\varepsilon}(Y_1),..,O_{r_m-\varepsilon}(Y_m)\}$, a set of obstacles
      $S = \{B_{R_1}(X_1),..,B_{R_n}(X_n)\}$
Result: A union of balls representing the collision space $\mathscr{C}^{col}(\mathscr{O}_\varepsilon^\phi)$
1 $\overline{GY_i} \leftarrow$ Centroid($\mathscr{O}_\varepsilon^\phi$)
2 $\mathscr{C}^{col}(\mathscr{O}_\varepsilon^\phi) \leftarrow \emptyset$
3 foreach $i \in \{1..m\},\ j \in \{1..n\}$ do
4     $\mathscr{C}^{col}(\mathscr{O}_\varepsilon^\phi) \leftarrow \mathscr{C}^{col}(\mathscr{O}_\varepsilon^\phi) \cup \mathrm{T}_{-\overline{GY_i}}(B_{R_j+r_i-\varepsilon}(X_j))$
5 end
6 return $\mathscr{C}^{col}(\mathscr{O}_\varepsilon^\phi)$
```

The construction of the collision space of the ε-core of the object is straightforward, see Alg. 1. Recall now that for a certain $U(\varepsilon,\phi)$ the collision space $\mathscr{C}^{col}(\mathscr{O}_\varepsilon^\phi)$ is contained within $\mathscr{C}^{col}(\mathscr{O}^\theta)$ for each $\theta \in U(\varepsilon,\phi)$, which gives us an approximation of the slice of $\mathscr{C}^{col}$ as $\mathscr{C}^{col}(\mathscr{O}_\varepsilon^\phi) \times U(\varepsilon,\phi)$.

4.3 Gluing The Slices Together

Once we have computed the collision space for each slice, we proceed by finding the connected components of the free space of each of them. As a result, we want to get s a $d-$dimensional simplicial complexes approximating the free space for each of the slices. For that, we use the approach proposed by McCarthy et al. in [10]. Since $\mathscr{C}^{col}(\mathscr{O}_\varepsilon^\phi)$ is a union of balls, it is convenient to represent it as an alpha complex – a discrete representation of the space. Alpha complexes are simplicial complexes which are particularly convenient for approximating unions of balls, as by the nerve theorem they preserve the homotopy type (and hence the necessary topological properties) of the corresponding union of balls. They have been used to approximate configuration spaces, for instance, in [7, 10]. We refer the reader to [3] for a complete and self-contained treatment of alpha complexes and the theory behind them. To approximate the free space, we compute a regular triangulation,

Algorithm 2: Glue-Slices

```
Data: A set of slices $\mathscr{C}_a^{col} = \{aSl_{U_1}^{col}, .., aSl_{U_s}^{col}\}$; their connected components
      $\{\{C_1^1, .., C_{n_1}^1\}, .., \{C_1^s, .., C_{n_s}^s\}\}$
Result: A connectivity graph $\mathscr{G}$
$\mathscr{G} \leftarrow \emptyset$
AddVertices($\mathscr{G}, \{C_1^1, .., C_{n_1}^1, .., C_1^s, .., C_{n_s}^s\}$
foreach $aSl_{U_i}^{col} \in \mathscr{C}_a^{col}$ do
    foreach $aSl_{U_j}^{col} \in$ Neighbours($aSl_{U_i}^{col}$) do
        foreach $C_k^i \in$ Components($aSl_{U_i}^{col}$) do
            foreach $C_l^j \in$ Components($aSl_{U_j}^{col}$) do
                if $C_k^i \cap C_l^j \neq \emptyset$ then
                    AddEdge($\mathscr{G}, (C_k^i, C_l^j)$)
                end
            end
        end
    end
end
return $\mathscr{G}$
```

which is a convex superset of the computed alpha complex, see [4]. We mark tetrahedra in the regular triangulation as “free” or “in collision”, and compute a set of connected components of the free space. We then mark the connected components as “bounded” and “unbounded”. Note that since both the object and the set of obstacles are bounded, the collision space is also bounded, and therefore the free space always has one unbounded connected component. We represent it by artificially adding an “infinite” tetrahedron.

Now when we have an approximation of the collision space as a union of slices

$$\mathscr{C}_a^{col} = \bigcup_{U_i, i \in \{1,..,s\}} aSl_{U_i}^{col}$$

and an alpha complex representing each slice aSl_U^{col}, we can study the relationship between slices. More precisely, we would like to build a connectivity graph $\mathscr{G}$, where the vertices represent the connected components $\{C_1^i, .., C_{n_i}^i\}$ of each slice U_i for $i \in \{1, ..s\}$, see Alg. 2.

Two vertices representing components $C_p \subset aSl_{U_i}^{col}$ and $C_q \subset aSl_{U_j}^{col}$, $i \neq j$, are connected by an edge if a direct transition between them is possible. That is, if *(i)* the sets U_i and U_j overlap, $U_i \cap U_j \neq \emptyset$, and *(ii)* the respective projections of the connected components to $\mathbb{R}^d$ intersect: $\mathrm{Pr}_{\mathbb{R}^d}(C_q) \cap \mathrm{Pr}_{\mathbb{R}^d}(C_p) \neq \emptyset$, where the projection operator is defined as $\mathrm{Pr}_{\mathbb{R}^d}(x, \phi) = x$. So, for each slice $aSl_{U_i}^{col}$, we consider its neighbours – i.e., slices $aSl_{U_j}^{col}$ such that $U_i \cap U_j \neq \emptyset$, and check whether the connected components of the free space of these pairs of slices (represented as d-dimensional simplicial complexes) overlap.

Once the connectivity graph is constructed, we can easily check if there is no path between two given configurations, and if a given configuration is caged. We run a depth-first search in $\mathscr{G}$ to compute its connected components, and then given two configurations c_s and c_g we simply check if the corresponding vertices of $\mathscr{G}$ belong to different connected components. If that is the case, there is no path between c_s and c_g.

Algorithm 3: Query-Connectivity

```
   **Data**: Two configurations, $c_i$ and $c_j$; a set of slices $\mathscr{C}_a^{col} = \{aSl_{U_1}^{col}, .., aSl_{U_s}^{col}\}$;
          their connected components $\{\{C_1^1, .., C_{n_1}^1\}, .., \{C_1^s, .., C_{n_s}^s\}\}$; the
          connectivity graph $\mathscr{G}$
   **Result**: Boolean, path existence
1  $aSl_i \leftarrow$ Slice-Containing-Configuration($c_i$, $\{aSl_{U_1}^{col}, .., aSl_{U_s}^{col}\}$)
2  $aSl_j \leftarrow$ Slice-Containing-Configuration($c_j$, $\{aSl_{U_1}^{col}, .., aSl_{U_s}^{col}\}$)
3  $C_i \leftarrow$ Component-Containing-Configuration($c_i$, $aSl_i$, $\{C_1^i, .., C_{n_1}^i\}$)
4  $C_j \leftarrow$ Component-Containing-Configuration($c_j$, $aSl_j$, $\{C_1^j, .., C_{n_1}^j\}$)
5  if Are-Connected($C_i$, $C_j$, $\mathscr{G}$) then
6      return Undefined
7  end
8  else
9      return False
10 end
```

For caging, we need to check whether we can reach an unbounded connected component starting from a given configuration, see Alg. 4. For each connected component of each of the slices, we know if it is bounded or not, each vertex of $\mathscr{G}$ has a corresponding mark. To prove caging, we check if the vertex of $\mathscr{G}$ corresponding to a certain configuration c is connected to an ”unbounded” vertex.

Algorithm 4: Check-Caging

```
   **Data**: A configuration $c_i$; a set of slices $\mathscr{C}_a^{col} = \{aSl_{U_1}^{col}, .., aSl_{U_s}^{col}\}$; their
          connected components $\{\{C_1^1, .., C_{n_1}^1\}, .., \{C_1^s, .., C_{n_s}^s\}\}$; the connectivity
          graph $\mathscr{G}$
   **Result**: Boolean, the configuration is caged
1  $aSl_i \leftarrow$ Slice-Containing-Configuration($c_i$, $\{aSl_{U_1}^{col}, .., aSl_{U_s}^{col}\}$)
2  $C_i \leftarrow$ Component-Containing-Configuration($c_i$, $aSl_i$, $\{C_1^i, .., C_{n_1}^i\}$)
3  if Connected-To-Unbounded($C$, $\mathscr{G}$) then
4      return Undefined
5  end
6  else
7      return True
8  end
```

5 Paths Construction

In this section, we discuss how our approximation of the configuration space can be used for path planning. Modern path planning algorithms, such as PRM, RRT, and their modifications, are guaranteed to find a path if one exists. However, they cannot prove that two configurations are disconnected. On the contrary, our approach is not guaranteed to find a path if one exists, as our approximation of the collision space is just a subset of the actual collision space. However, we can prove path non-existence. Thus, path planning algorithms and our method can be considered as complementary techniques and can be run in parallel to solve path planning problem more efficiently. Furthermore, our approximation of the collision space can be used to construct path candidates which are not guaranteed to be collision-free, but can be used as initial approximations and to be further improved using some path optimization technique, such as CHOMP [19]. This is especially useful in cluttered environments, when we need to find a path through a narrow passage. Our approach can be seen as a continuation of the pioneering work by Basch et al., [2].

Since our approximation of the collision space is not complete, we are limited to a certain class of paths which we can construct. We call them *decomposable paths*: – collision-free paths that can be decomposed into finitely many motions in $SO(d)$ followed by a motion in $\mathbb{R}^d$. Formally, we define it as follows:

Definition 3 *A path $\gamma : [0,1] \to SO(d)$ is decomposable if there exists a family of paths $\gamma_1, \gamma_1', .., \gamma_n, \gamma_n'$ such that $\mathrm{Pr}_{\mathbb{R}^d}(\mathrm{Im}(\gamma_i))$ and $\mathrm{Pr}_{SO_d}(\mathrm{Im}(\gamma_i'))$ are constant paths in $\mathbb{R}^d$ and $SO(d)$ respectively*[3]*, and $\gamma = \gamma_1 * \gamma_1' * .. * \gamma_n * \gamma_n'$, where by $\gamma_i * \gamma_j$ we mean a composition of paths γ_i and γ_j.*

However, this class is nevertheless useful as we can show that if there exists some collision-free path in the free space, our approximation of the configuration space allows us to construct a decomposable approximation. Namely, instead of considering the original object, we construct a decomposable path for its ε-core, and we show that we can always do that providing there is a collision-free path:

Lemma 1 *If two configurations c and c' of the original object $\mathscr{O}$ are connected, then there is a decomposable path between the respective configurations of its ε-core $\mathscr{O}_\varepsilon$.*

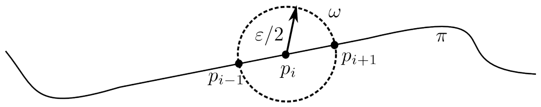

Fig. 5 The arcs $(p_{i-1}, p_i$ and $(p_i, p_{i+1}$ of the projection π are completely contained within a ball ω of radius $\varepsilon/2$

[3] Here $\mathrm{Pr}_{\mathbb{R}^d}(.)$ and $\mathrm{Pr}_{SO(d)}(.)$ denote the projections to $\mathbb{R}^d$ and $SO(d)$ respectively, and $\mathrm{Im}(\gamma)$ denotes the image of γ in $SO(d)$.

Proof. If there exists a decomposable path between c and c' for the original object $\mathscr{O}$, then the same path is valid for $\mathscr{O}_\varepsilon \subset \mathscr{O}$. Otherwise, let us pick any non-decomposable path γ between c and c'. Note that along this path $\mathscr{O}_\varepsilon$ is always at least at a distance ε from the obstacles. Let us now explicitly decompose γ by choosing waypoints $c = (p_0, r_0), \ldots, (p_n, r_n) = c'$ from γ and constructing $\gamma' = \{(p_0, r_0), (p_1, r_0), (p_1, r_1), \ldots, (p_{n-1}, r_{n-1}), (p_n, r_{n-1}), (p_n, r_n)\}$, where p_i are the coordinates of the $\mathbb{R}^d$ component of $SE(d)$ and r_i are the coordinates of the $SO(d)$ component. The resulting decomposition will consist of alternating pure translation and pure rotation paths sending the point (p_i, r_i) to (p_{i+1}, r_i) through a translation-only segment and then to (p_{i+1}, r_{i+1}) through a rotation-only segment.

Assume the waypoints $c = (p_0, r_0), \ldots, (p_i, r_i)$ are already chosen. Let $p = \mathrm{Pr}_{\mathbb{R}^d}(\gamma)$ be the projection of γ onto $\mathbb{R}^d$. We choose the next point p_{i+1} as the second endpoint of the longest arc (p_i, y) of the path π which is fully contained within an $\varepsilon/2$-ball ω centered at p_i (or as the $\mathbb{R}^d$-component of c', in case the part of π connecting it to p_i lies inside ω), and make sure that p_{i+1} does not coincide with p_{i-1}.

Let us show that a pure translation of $\mathscr{O}_\varepsilon$ from (p_i, r_i) to (p_{i+1}, r_i) is collision-free. Since π is a projection of a collision-free path, $\mathscr{O}$ in configuration (p_i, r_i) does not collide with obstacles. Then any point of $\mathscr{O}_\varepsilon$ in (p_i, r_i) is separated at least at a distance ε from the obstacles. Pure translation to (p_{i+1}, r_i) will move each point of $\mathscr{O}_\varepsilon$ at distance $\varepsilon/2 < \varepsilon$, so it will not collide with the obstacles.

Now consider a pure rotation of $\mathscr{O}_\varepsilon$ from (p_{i+1}, r_i) to (p_{i+1}, r_{i+1}). Assume that there is some $r' \in [r_i, r_{i+1}]$ such that $\mathscr{O}_\varepsilon$ in configuration (p_{i+1}, r') collides with obstacles. Given the continuity of the path between (p_i, r_i) and (p_{i+1}, r_{i+1}), there is some configuration (p', r') such that $p' \in [p_i, p_{i+1}]$. Since p_{i+1} is chosen in a way that the arc (p_i, p_{i+1}) lies inside ω, p' also lies inside ω. Then the distance between p' and p_{i+1} is less than ε. Therefore, since (p', r') is collision-free for the original object $\mathscr{O}$, (p_{i+1}, r') is collision-free for $\mathscr{O}_\varepsilon$, which contradicts our assumption.

Finally, we note that using Lemma 1 we can extend Algorithm 3 to provide an approximate path between two configurations (p, r) and (p', r') if one exists. To do this, we can have the algorithm output a sequence of pairs $(C_1, S_1), \ldots, (C_n, S_n)$ which comprise the path in the connectivity graph $\mathscr{G}$, where the C_i are the components and the S_i. To produce a path we start from the configuration $(p, r) \in C_1 \times S_1$ and use a local planner to find a path to some point $(p_2, r) \in (C_1 \cap C_2) \times S_1$ and then we rotate this configuration to (p_2, r_2) where $r_2 \in S_2$. We proceed in this manner until we reach $(p_n, r_n) \in C_n \times S_n$, since by construction $(p', r') \in C_n \times S_n$, we can use a local planner to find a path from (p_n, r_n) to (p', r_n) and finally rotate the final configuration to (p', r').

6 Discussion and Possible Extensions

In this paper, we propose an approach towards proving caging and path non-existence for rigid objects in 2D and 3D workspaces. We compute an approximation of the collision space of the object, represent it as a collection of lower dimensional projections, and analyze the connectivity of the free space of the object. We also discuss how our approach can be generalized to arbitrary configuration spaces.

Our approach can be potentially extended to a broader class of configuration spaces, for example, in order to prove path non-existence for an articulated robot. Indeed, as long as the configuration space can be represented as a product $\mathscr{C} = \mathscr{C}_1 \times \mathscr{C}_2$, where $\mathscr{C}_1 \subset \mathbb{R}^d$ and $\mathscr{C}_2$ is a compact set, we can construct a finite open cover $\mathscr{C}_2 = \bigcup_{U_i \subset \mathscr{C}_2} U_i$, and for each slice $Sl_i = \mathscr{C}_1 \times U_i$ construct collision space approximations using an ε-core of the robot.

References

[1] Barraquand, J., Kavraki, L., Latombe, J.-C., Motwani, R., Li, T.-Y., Raghavan, P.: A random sampling scheme for path planning. In: The International Journal of Robotics Research, 16(6), pp. 759774 (1997).

[2] Basch, J., Guibas, L. J., Hsu, D., Nguyen, A. T.: Disconnection proofs for motion planning. In: IEEE International Conference on Robotics and Automation (2001), pp. 1765-1772.

[3] Edelsbrunner, H., Harer, J.: Computational topology: an introduction. American Mathematical Soc., 2010.

[4] Edelsbrunner, H.: Weighted alpha shapes. University of Illinois at Urbana-Champaign, Department of Computer Science, 1992.

[5] Kuperberg, W.: Problems on polytopes and convex sets. In: DIMACS Workshop on polytopes (1990), pp. 584-589.

[6] Latombe, J.-C.: Robot Motion Planning. Norwell, MA, USA: Kluwer Academic Publishers (1991).

[7] Mahler, J., Pokorny, F. T., McCarthy, Z., van der Stappen, A. F., Goldberg, K.: Energy-bounded caging: Formal definition and 2-D energy lower bound algorithm based on weighted alpha shapes. In: IEEE Robotics and Automation Letters, 1(1), pp.508-515 (2016).

[8] Makita, S., Maeda, Y.: 3D multifingered caging: Basic formulation and planning. In: IEEE Intelligent Robots and Systems (2008), pp. 2697–2702.

[9] Makita, S., Okita, K., Maeda, Y.: 3D two-fingered caging for two types of objects: sufficient conditions and planning. In: International Journal of Mechatronics and Automation, 3(4), pp. 263-277 (2013)

[10] McCarthy, Z., Bretl, T., Hutchinson, S.: Proving path non-existence using sampling and alpha shapes. In: IEEE International Conference on Robotics and Automation (2012), pp. 2563-2569.

[11] Pipattanasomporn, P., Sudsang, A.: Two-finger caging of concave polygon. In: IEEE International Conference on Robotics and Automation (2006), pp. 2137–2142.

[12] Pipattanasomporn, P., Sudsang, A.: Two-finger caging of nonconvex polytopes. In: IEEE Transactions on Robotics, 27(2), pp. 324-333 (2011).

[13] Pokorny, F. T., Stork, J. A., Kragic, D.: Grasping objects with holes: A topological approach. In: IEEE International Conference on Robotics and Automation (2013), pp. 1100–1107.

[14] Rimon, E., Blake, A.: Caging planar bodies by one-parameter two-fingered gripping systems. In: The International Journal of Robotics Research, 18(3), pp. 299–318 (1999).

[15] Rodriguez, A., Mason, M. T., Ferry, S.: From caging to grasping. In: The International Journal of Robotics Research, 31(7), pp. 886-900 (2012).

[16] Stork, J. A., Pokorny, F. T., Kragic, D.: Integrated Motion and Clasp Planning with Virtual Linking. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (2013), pp. 3007-3014.

[17] Stork, J. A., Pokorny, F. T., Kragic, D.: A Topology-based Object Representation for Clasping, Latching and Hooking. In: IEEE-RAS International Conference on Humanoid Robots (2013), pp. 138-145.

[18] Pereira, G. A. S., Campos, M.F.M., Kumar, V.: Decentralized algorithms for multi-robot manipulation via caging. In: The International Journal of Robotics Research 23(7-8), pp. 783 – 795 (2004).

[19] Ratliff, N., Zucker, M., Bagnell, J. A., Srinivasa, S.: CHOMP: Gradient optimization techniques for efficient motion planning. In: IEEE International Conference on Robotics and Automation (2009), pp. 489-494.

[20] Varava, A., Kragic, D., Pokorny, F. T.: Caging Grasps of Rigid and Partially Deformable 3-D Objects With Double Fork and Neck Features. In IEEE Transactions on Robotics, 32(6), pp. 1479-1497 (2016).

[21] Vahedi, M., van der Stappen, A. F.: Caging polygons with two and three fingers. In: The International Journal of Robotics Research, 27(11-12) pp. 1308–1324 (2008).

[22] Zhang, L., Young, J. K., Manocha, D.: Efficient cell labelling and path non-existence computation using C-obstacle query. In: The International Journal of Robotics Research, 27(11-12), pp. 1246-1257 (2008).