Rope Caging and Grasping

Tsz-Ho Kwok[1], Weiwei Wan[2], Jia Pan[3], Charlie C.L. Wang[4†], Jianjun Yuan[5], Kensuke Harada[2] and Yong Chen[1]

***Abstract*— We present a novel method for caging grasps in this paper by stretching ropes on the surface of a 3D object. Both topology and shape of a model to be grasped has been considered in our approach. Our algorithm can guarantee generating local minimal rings on every topological branches of a given model with the help of a Reeb graph. Cages and grasps can then be computed from these rings, and physical experimental tests have been conducted to verify the robustness of our approach.**

I. Introduction

Caging, as compared to grasping, is non-dynamic, topological, and therefore robust to uncertainty. Given the noisy point clouds or the noisy results of model-based estimation, caging can find finger positions or configurations that constrain the target. The target may move inside the cage formed by the fingers, but it will never escape from the cage. To cage an object, we do not need to know its physical properties like mass and inertia matrix.

Despite the advantages, caging has a major disadvantage that the caged target is not immobilized and its exact position and orientation are unknown. Caging can be used to catch and constrain something, but it is difficult to be used to assemble components or operate tools.

A. Motivation

The solution to overcome these disadvantages is grasping by caging. For example, the workers in Fig.1 are picking and placing wine port using caging. In the first step, they drop a circular rope and wrap it around the thinner part of the port. The port is caged by the rope after the actions of dropping and wrapping. In the second step, they stretch the circular rope by lifting it up and make the rope firmly in contact with the port's surface. While the first step only cages the port and the port is not immobilized, the second step grasps the object and enables the worker to place the port down to an expected position.

Motivated by this example, in this paper we propose a grasping by caging approach based on ropes. The seminal work of grasping by caging was done by Rodriguez [1] and Wan [2]. Rodriguez extended the idea of caging and proposed F-caging. Given a finger formation, Wan, on the other hand, recovered its configuration space and computed the optimized configuration that could cage an object and at the same time could shrink to grasping. Both papers proposed interesting ideas but neither demonstrated successful applications. Rodriguez's work was more theoretical than practical, and is more about a concept than some rigid-form solutions. Wan's work is applicable to 2D objects and was demonstrated with 2D pick-and-place and multi-robot transportation.

Unlike these works, our approach concentrates on ropes and studies using rope to cage and grasp 3D objects with slim waists or with topological holes. We employ reeb graph to extract the position of slim waists or the holes of object models, initialized some caging rings using the extracted mesh segments, and estimate the stable grasps by stretching the initial rings like a tapeline. We ensures caging by requiring a finger ring be larger than a stretched caging ring, and estimates the object's position and orientation by assuming that the grasping from this caging will converge to the stretched caging ring. Our work can be used to plan the grasps of rope or rope-like end-effectors. We demonstrate the performance of our method with both a rope and a soft gripper in the experimental tests.

B. Related Work

Grasping research in literature can be classified into two major groups: force-closure and caging. Earlier works are mainly based on the force-closure methods (e.g., state-of-the-art simulators such as GraspIT and OpenGRASP). Recently, approaches to realize stable grasping by a caging grasp become more and more popular (ref. [1], [2]).

An interesting work is to use caging grasp to manipulate articulated objects with handles [3]. With the help of computing a topological loop on a given surface [4], Stork et al. [5] can plan motions for clasping on objects with handles. Our approach presented in this paper is more general – an object to be grasped by our rope caging is not

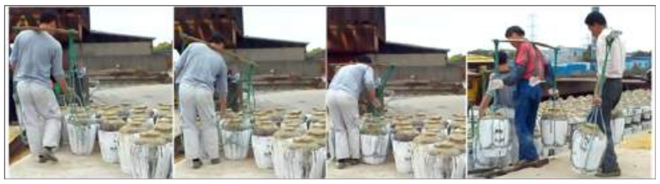


Fig. 1. With the help of ropes, objects can be stably caged in real world. Our work is motivated by this caging method with a long history.

The authors thank Zhongfeng Qian and Zheng Wang from the University of Hong Kong, and Hao Zhang from Dorabot Inc. for assistance with the design and fabrication of the soft gripper used in the experiment.

[1]T. Kwok and Y. Chen are with the Epstein Department of Industrial and Systems Engineering, University of Southern California. [2]W. Wan and K. Harada are with National Inst. of AIST. [3]J. Pan is with the Department of Computer Science, University of Hong Kong. [4]C. Wang is with the Department of Design Engineering, Delft University of Technology; [5]J. Yuan is with the Robotics Institute, Shanghai Jiaotong University

[†]Corresponding Author; Email: `c.c.wang@tudelft.nl`.

necessary to have a handle (e.g., the Venus model shown in Fig.6 and the spray container model shown in Fig.9(a)). Recently, more and more researches in robotics start to employ the geometric information of a 3D object. Curvature-based feature has been used by Calli et al. [6] to identify most proper place for grasping. They used concave points of a 2D elliptic Fourier descriptor of an object. Gaussian curvature is employed in [7] to conduct the segmentation of object for the grasping analysis. Tsuji et al. [8] modeled objects with multiple quadric surfaces and generated grasping posture by selecting the constricted parts. Zarubin et al. [9] proposed an idea of using geodesic balls on the object's surface in order to approximate the maximal contact surface between a grasp and an object. Two types of caging grasps are developed: circle caging and sphere caging, where circle caging is similar to our rope caging. However, unlike our approach, their method does not provide the capability of capturing either the topological extremity such as handle or the geometric extremity like bottle-neck shape.

Reeb graph is a topological analysis tool that was previously employed to conduct a variety of applications in computer vision and graphics, including model decomposition [10], [11], parameterization [12] and shape matching [13]. In our approach, its concept is borrowed to analyze whether a topological branch on the given object is missed when constructing rings by uniformly sampling a scalar field. An algorithm is developed to ensure every branch has one ring. Note that, we do not need to construct the graph representation but only identify the critical points on a Reeb graph in our approach. Details can be found in Section III-A and III-B.

C. Contribution

The technical contribution of our work presented in this paper can be summarized as follows.

- We introduce the idea of stretching ropes on the surface of a 3D object to analyze the possibility of caging the model using a ring. Both topology and shape of a model to be grasped have been considered.
- A novel algorithm has been developed to generate minimal rings on every branches of the given model, which is guaranteed by the topology analysis with the help of a Reeb graph.
- Our approach has been evaluated in both simulation and physical experimental tests, where a rope and a soft gripper are employed in experimental tests.

Although the topology analysis based on a topological loop has been conducted in prior work (e.g., [5]) for grasping, to the best of our knowledge, this is the first approach that considers the minimal rings on all handles and branches.

Not only the topology but also the geometry information have been considered in our approach. As a result, the rope caging technique developed in this paper can be applied to models with very simple topology (e.g., genus *zero* models) if a local shape extremity can be touched by the elastic rope, which is similar to the real situation as shown in Fig.1. Cages and grasps can be computed from the rings generated by our algorithm (Section IV). Physical experimental tests have been taken to verify the validity of this approach on models with different shape and topology (Section V).

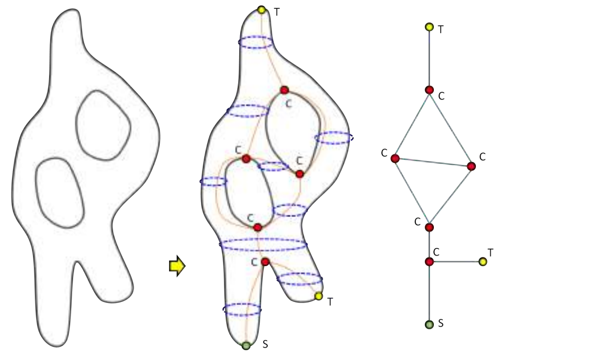


Fig. 2. Reeb graph of a given 3D model can be obtained by tracking the evolution of the level sets of a real-valued function on its surface. The green node is a source point (with minimal function value), the yellow nodes are terminal points (with local maximal function value), and the red points are called critical points with the topology change of iso-curves – *splitting* or *merging*. The portion between these points are classified into different branches (see the light-blue segments in the right picture). Our algorithm for generating initial ropes can ensure missing no branch on the Reeb graph. Cross-sectional rings are generated for each branch in our algorithm (see the blue rings in dashed line in the middle picture).

II. Background

We assume each object to be caged has a corresponding 3D model represented by a polygonal mesh in the computer system. Following the notation in [14], a polygonal mesh $\mathcal{M}$ is defined as a pair $(\mathcal{K}, \mathcal{V})$ where $\mathcal{K}$ is a complex specifying the connectivity of vertices, edges and faces (i.e., the topological graph of $\mathcal{M}$) and $\mathcal{V} = \{\mathbf{v}_1, \cdots, \mathbf{v}_n\}$ is the set of vertices defining the shape of polyhedral surface in $\Re^3$. To simplify the implementation, every polygonal face in $\mathcal{M}$ is subdivided into triangles. From $\mathcal{K}$, it is very easy for our algorithm to get the adjacent vertices, edges and faces of a vertex in constant time; the same, the left/right faces of an edge and the three nodes/edges of a triangle can also be obtained by constant time cost. The geometric coverage of our method is confined to the domain of *two-manifold* polyhedral surface. Specifically, for every point on the surface of a two-manifold, there exists a sufficiently small neighborhood that is topologically the same as an open disk in $\Re^2$. Such representation of a real object can be obtained via 3D reconstruction (e.g., KinectFusion [15]). In our approach, elastic ropes will be constructed and then actively moved (i.e., shrunk) on a mesh model $\mathcal{M}$. Each rope is stored as an ordered list of surface points as $\mathcal{L} = \{\mathbf{a}_1, \cdots, \mathbf{a}_m\}$, where a surface point could be located on an edge or inside a face of $\mathcal{M}$. They are stored as attribute curves attached on $\mathcal{M}$. Details can be found in [16]. This type of data structure can prevent pulling/pushing an elastic rope during the process of 'stretching' (i.e., $\mathcal{L} \subset \mathcal{M}$ is ensured). The time-consuming step of collision detection can be avoided.

A Reeb graph [17] is a fundamental structure that encodes

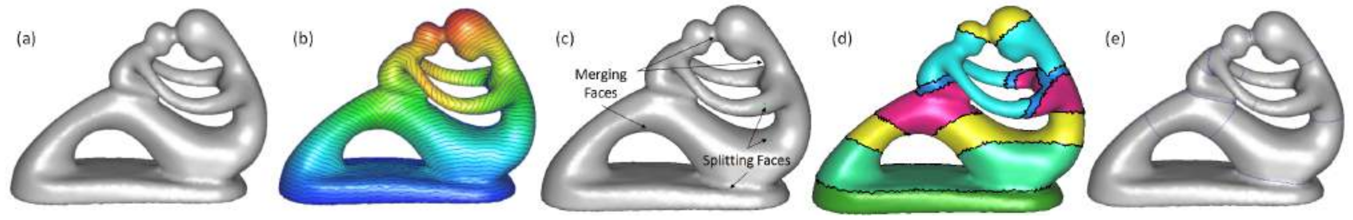


Fig. 3. An overview of computing local minimal rings for caging and grasping. (a) The input 3D model. (b) A scalar field for topology analysis is computed based on geodesic distances, and the distribution of field values is illustrated by a color map with blue and red denoting minimal and maximal values respectively. Isocurves are also displayed, which can be used as initial rings (Section III-B). (c) Topology analysis is conducted by the scalar field and the critical points can be found by tracking merging/splitting faces (Section III-A). (d) Surface of the given model can be subdivided by the critical points into a few topological branches (displayed in different colors). (e) The initial rings are stretched on the surface and the candidates for caging/grasping can be obtained – see the blue rings (Section III-C).

the topology of a shape, which can be obtained by tracking the evolution of the level sets of a real-valued function f on a manifold. Given a 3D model $\mathcal{M}$, its topology can be represented by a Reeb graph with nodes denoting the change of topology and all surface regions on the same branch sharing the same topology (see Fig.2 for an illustration). Every node on the Reeb graph corresponds to the critical level sets $f^{-1}(c)$. The pattern in which a branch meets a critical point reflects the change in topology of the level set $f^{-1}(t)$ as t passes through the critical value c. By tracing all the critical points on a Reeb graph, all branches of a given model can be trimmed out by the level sets according to the critical values. The basic idea of our rope caging/grasping approach is to generate a few elastic ropes on each branch (see the blue cross-sectional rings in Fig.2) and stretch them into the minimal length. The resultant rings on the surface of a given model will be employed to supervise the caging and thereafter grasping. The analysis based on Reeb graph can help us ensure no topological branch is missed when constructing the initial rings.

III. Methodology

This section presents the methodology of computing local minimal rings for rope caging and grasping. The topology analysis that guarantees not missing any branch of a given model $\mathcal{M}$ is taken with the help of a geodesic distance field $\mathcal{G}_{\mathcal{M}}$. Critical points are found by a flooding algorithm. With the help of critical points, initial rings are constructed for each branch and thereafter stretched into local minimal rings. Figure 3 shows an overview of steps in our algorithm.

A. Scalar Field Based Topology Analysis

Topology analysis is taken on the surface of a given model $\mathcal{M}$ to avoid missing any topological branch when generating initial rings. The concept of Reeb graph is employed in our analysis; however, we do not really construct a Reeb graph but only find out its critical points on $\mathcal{M}$.

First of all, a scalar field $\mathcal{G}_{\mathcal{M}}$ is constructed on $\mathcal{M}$. One simple option as used in [13] is the height field of a model. However, it can easily miss the handles that are nearly orthogonal to the z-axis. Instead, *Approximate Geodesic Distance* (AGD) field is employed in our approach. Assuming $\mathcal{M}$ has been properly oriented, all mesh vertices that are

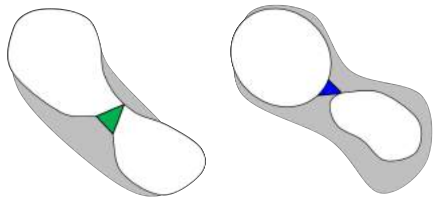


Fig. 4. Two types of critical faces: (left) a splitting face in green and (right) a merging face in blue, where the flooded region is displayed in gray.

on the ground are located and assigned as sources of AGD. $\mathcal{G}_{\mathcal{M}}(\mathbf{p})$ $(\forall \mathbf{p} \in \mathcal{M})$ is then computed according to these sources. In our implementation, a Dijkstra's algorithm with virtual paths is employed (ref. [18]). More precise methods such as [19], [20] can also be used here; however, $\mathcal{G}_{\mathcal{M}}$ in our approach is not necessary to be precise as it is only for the purpose of topology analysis.

After computing $\mathcal{G}_{\mathcal{M}}$, we conduct a flooding algorithm akin to [10] to find out the critical points on $\mathcal{M}$. The $\mathcal{G}_{\mathcal{M}}$ score of a face is defined as the maximal value of $\mathcal{G}_{\mathcal{M}}$ on its three vertices. A flooded region $\mathcal{F}$ is first initialized by faces touched the ground. Other faces are then sorted by their scores. Starting from the face with lowest score, faces are added into $\mathcal{F}$ one by one. Every time when a new face f is inserted, we check the topology of $\mathcal{F}$'s boundary, $\partial\mathcal{F}$, by counting the number of loops in $\partial\mathcal{F}$ – denoted by $|\partial\mathcal{F}|$.

- $|\partial(\mathcal{F} \bigcup \{f\})| > |\partial\mathcal{F}|$: f is a *splitting* face;
- $|\partial(\mathcal{F} \bigcup \{f\})| < |\partial\mathcal{F}|$: f is a *merging* face;
- $|\partial(\mathcal{F} \bigcup \{f\})| = |\partial\mathcal{F}|$: f is *not* critical.

Both splitting and merging faces are considered as critical (see the illustration in Fig.4). In a critical face, the vertex with largest field value is recorded as a critical point. When three edges of a critical face are all on the boundary of $\mathcal{F}$ before inserting it, the face is located as the tail of $\mathcal{M}$. A critical point on this type of face is called T-critical, and other type of critical points are called C-critical as illustrated in Fig.2.

B. Computing Initial Rings

In this step, a finite set of candidate rings are generated on the model. Each ring is constructed according to an isovalue. Specifically, for generating rings for $\mathcal{G}_{\mathcal{M}}(\mathbf{p}) = c$, the isocurve consists of line segments extracted on the surface of $\mathcal{M}$ by first searching faces with vertex scores both smaller and greater than c and then producing a line segment intersect two edges in each of these faces. The endpoints of a line segment must be located on the edges with its two endpoints greater and smaller than c. The position of intersection is determined by the linear interpolation. Using the local connectivity of $\mathcal{M}$ stored in $\mathcal{K}$, the line segments can be linked up into one or a few loops.

Given a maximal field value $g_{\max}$ on $\mathcal{G}_{\mathcal{M}}$, the simplest method to generate rings is by isocurves with uniformly sampled k isovalues. Specifically, isocurves

$$\mathcal{G}_{\mathcal{M}}(\mathbf{p}) = \frac{i}{k+1} g_{\max} \quad (i = 1, \cdots, k) \tag{1}$$

are extracted to form initial rings uniformly (as shown in Fig.3(b)). Rings on an isocurve are generated by the aforementioned method. However, this simple method may miss some topological branches on a given model with complex topology (e.g., when a handle happens in the region between $\mathcal{G}_{\mathcal{M}}(\mathbf{p}) = \frac{i}{k+1} g_{\max}$ and $\mathcal{G}_{\mathcal{M}}(\mathbf{p}) = \frac{i+1}{k+1} g_{\max}$). A more sophisticated algorithm is developed below to avoid such scenarios.

After constructed rings by uniformly sampled isovalues (e.g., $k = 50$ is employed in all our tests), rings are inserted on those missed branches. We build a list $\mathcal{C}$ by the field values on all critical points in an ascending order. For example, when a model has n critical points found in the topology analysis step, the list is like

$$\mathcal{C} = \{0, c_1, c_2, \cdots, c_n\} \ (\forall i, \ c_i < c_{i+1}).$$

Note that both C-type and T-type critical points are included in this list. The last critical point is a T-critical point, $c_n = g_{\max}$. Defining $\tau = \frac{k+1}{g_{\max}}$, we then check if $\lfloor \tau c_i \rfloor = \lfloor \tau c_{i+1} \rfloor$. If this happens, no ring has been constructed on the branches between c_i and c_{i+1} by uniform sampling. We insert a ring at the middle of these two critical points. Specifically, initial rings are constructed on the following isocurves

$$\mathcal{G}_{\mathcal{M}}(\mathbf{p}) = \frac{1}{2}(c_i + c_{i+1}) \quad (\forall_i \ \lfloor \tau c_i \rfloor = \lfloor \tau c_{i+1} \rfloor) \tag{2}$$

These initial rings will be further stretched on the surface of $\mathcal{M}$ to catch the local minimal of its shape.

C. Stretching for Local Minimal Rings

In this step of our algorithm, each initial ring is iteratively shortened on $\mathcal{M}$ until reaching a local minimum. As a result, rings generated by this method can capture the features such as bottle necks and bumpy regions and use them to cage an object to be grasped. These analogies learned from realistic lead to a smart method for caging/grasping 3D objects.

For a rope $\mathcal{L}$ stored as a list of surface points $\{\mathbf{a}_1, \cdots, \mathbf{a}_m\}$, the method how it is constructed ensures that

1) all points are located on the edges (as the rings/ropes are generated from isocurves);
2) the segment between two neighboring points lies on the same triangle (i.e., there is no segment across an edge).

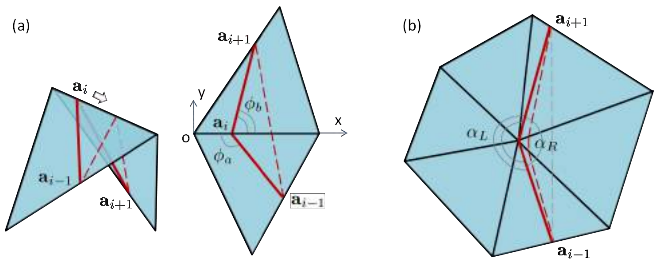


Fig. 5. Two types of operators for local stretching: (a) an edge operator and (b) a node operator, where the bold lines in red show the position of a rope before local stretching.

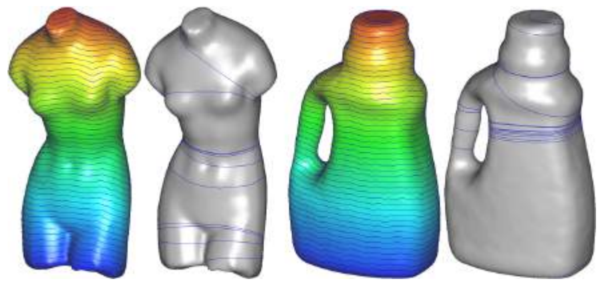


Fig. 6. Local minimal rings generated on two examples: (left) the Venus model with 6.4k vertices – 11 rings are generated, and (right) a container model with 10k vertices — results in 19 curves.

These nice properties of a rope's representation lead to a simple algorithm for stretching $\mathcal{L}$ on $\mathcal{M}$.

Points in $\mathcal{L}$ are moved one by one to meet an *equal angle condition* [16]. As shown in Fig.5(a), the optimal position of point $\mathbf{a}_i$ on the edge that results in a minimal length of $\|\mathbf{a}_{i-1}\mathbf{a}_i\| + \|\mathbf{a}_i\mathbf{a}_{i+1}\|$ is a point that makes $\phi_a = \phi_b$. This can be easily proved by flattening two triangles adjacent to the edge onto a plane together with the points. An operator pushing points of a rope to move on edges of $\mathcal{M}$ is called *edge operator*.

More complicated scenario may happen after applying edge operators for a few iterations – quite a few very short segments crowded around a mesh vertex (see Fig.5(b)). Then, the rope is pushed to either the left (when the left surface angle α_L is smaller than the right surface angle α_R) or the right (if $\alpha_L > \alpha_R$). Such an operator is called *node operator*. Again, the purpose of this operator is to make $\mathcal{L}$ satisfy the *equal angle condition*. Detail analysis and the implementation of these two operators can be found in [16].

The edge operator and the node operator are repeatedly applied to each rope initially constructed from isocurves. The iteration is stopped when no point on the rope can be moved any more. When all points on a rope have been located in the same triangle, this rope is removed from candidate rings as it indicates that the given model cannot be caged by this rope. Lastly, overlapped ropes stretched from different initial positions are merged to remove redundancy. Examples can be found in Fig.6. The resultant ropes are the candidates to be used in caging and grasping. Each rope is sampled into a set of points together with their surface normals, which are passed to the grasper controller to take further action.

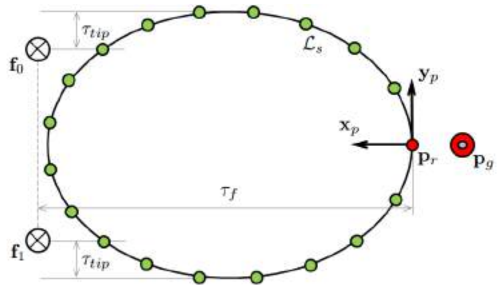


Fig. 7. An illustration of computing the positions of finger-tips, $\mathbf{f}_0$ and $\mathbf{f}_1$, for caging.

IV. Computing the Grasps

Algorithm 1: Computing the caging grasp using the stretched ring.

```
Data: a mesh model $\mathcal{M}$ and a local minimal ring $\mathcal{L}_s$
Result: A list of triples $\mathcal{G}_{\mathcal{L}_s}$ where each element
        indicates the position and configuration of the
        caging gripper: ($\mathbf{p}_g$, $\mathbf{R}_g$)
($\mathcal{P}$,$\mathcal{N}$) ← SampleRing($\mathcal{L}_s$);
for each $(\mathbf{p}_r, \mathbf{n}_r) \in (\mathcal{P},\mathcal{N})$ do
    $\mathbf{x}_p \leftarrow \mathbf{n}_r$;
    $\mathbf{z}_p$ ← CrossProduct($\mathbf{n}_{r_{next}}$,$\mathbf{n}_r$);
    $\mathbf{y}_p$ ← CrossProduct($\mathbf{z}_p$,$\mathbf{x}_p$);
    $\mathbf{R}_p \leftarrow [\mathbf{x}_p; \mathbf{y}_p; \mathbf{z}_p]$;
    $\{\mathbf{p}_g, \mathbf{R}_g, \mathbf{f}\}$
        ←ComputeHandConf($\mathbf{p}_r$, $\mathbf{R}_p$, $\mathcal{P}$, $\mathcal{N}$);
    if NOT IsColliding($\{\mathbf{p}_g, \mathbf{R}_g, \mathbf{f}\}$,$\mathcal{M}$) then
        $\mathcal{G}_{\mathcal{L}_s} \leftarrow \{\mathbf{p}_g, \mathbf{R}_g\}$;
```

To determine the grasps, each local minimal ring is sampled evenly to compute the approaching vectors and orientations of a caging hand. Let $\mathcal{P}$ and $\mathcal{N}$ denote the set of sampled points and their corresponding normals. Their elements are $\mathbf{p}_r$ and $\mathbf{n}_r$. A local frame $[\mathbf{x}_p; \mathbf{y}_p; \mathbf{z}_p]$ is constructed at each sample point to let $\mathbf{x}_p$ and $\mathbf{y}_p$ align with the normal and the tangent of the ring respectively. We use them together with $\mathbf{p}_r$ and all other points on the sampled ring to compute the position, the orientation, and the finger-tip positions $\mathbf{f}$ of the hand. This is implemented in the `ComputeHandConf` function as follows.

- First, $\mathbf{R}_p$ is directly used as $\mathbf{R}_g$ – that is the orientation of the hand.
- Second, the position of the hand is assigned as

$$\mathbf{p}_g = \mathbf{p}_r - \tau_g \mathbf{x}_p, \tag{3}$$

  where τ_g is a threshold gap between the hand and the sample point. Note that we assume all normal vectors sampled from a mesh model pointing inwards.
- Lastly, the positions of finger tips, $\mathbf{f}$, are computed. The vector consists of $\mathbf{f}_0$ and $\mathbf{f}_1$, which presents positions of the first and the second finger-tips respectively. They can be computed by

$$\mathbf{f}_i = \mathbf{p}_r + \tau_f \mathbf{x}_p + \tau_i \mathbf{y}_p \quad (i = 0, 1), \tag{4}$$

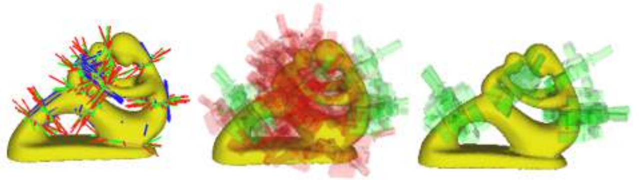
  where the threshold τ_f is assigned with a value smaller than the length of the fingers to ensure caging.

$$\tau_0 = \max_{\forall \mathbf{p}_p \in \mathcal{P}} ((\mathbf{p}_p - \mathbf{p}_r) \cdot \mathbf{y}_r) - \tau_{tip} \tag{5}$$

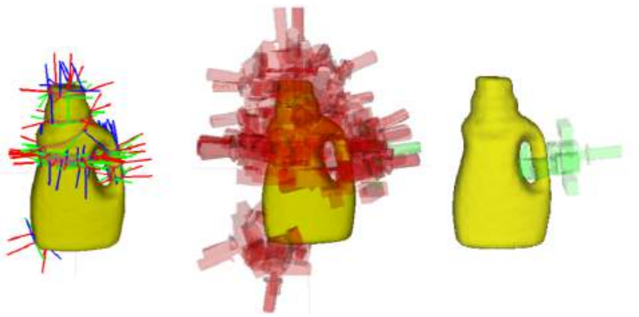
$$\tau_1 = \min_{\forall \mathbf{p}_p \in \mathcal{P}} ((\mathbf{p}_p - \mathbf{p}_r) \cdot \mathbf{y}_r) + \tau_{tip} \tag{6}$$

  τ_{tip} is a fixed value to ensure the closure. τ_0 and τ_1 make the distance between two finger-tips smaller than the diameter of $\mathcal{L}_s$ along the direction of $\mathbf{y}_p$ and therefore guarantee a successful cage.

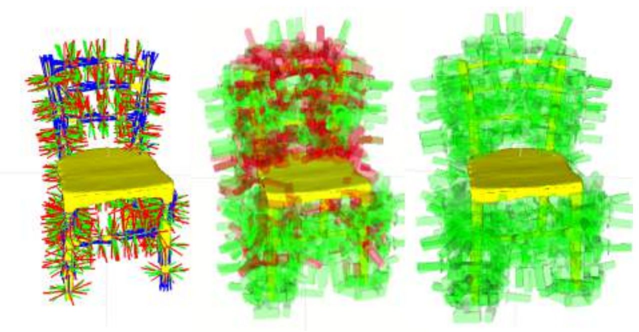
An illustration about the positions of hand and finger-tips with reference a local minimal ring is given in Fig.7. After computing the configuration of a hand for grasping, we further pass it to a collision checking function, `IsColliding`, to ensure caging by the hand at this ring is collision-free. Only the collision-free caging grasps are saved as results.



(a) Computing the caging grasps of a fertility model



(b) Computing the caging grasps of a detergent model



(c) Computing the caging grasps of a chair model

Fig. 8. We sample each stretched ring evenly to compute the approaching vectors and orientations of a caging hand. Caging test together with collision detection are then performed to ensure robustness of the results.

Fig.8 shows the results of three examples. The left image of each sub-figure shows the local minimal rings, their

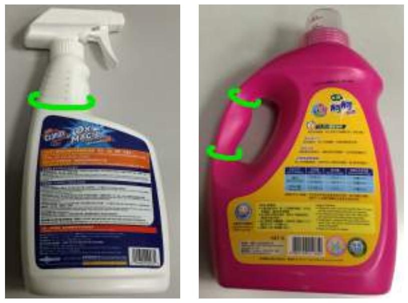

(a) Spray container (b) Detergent container

Fig. 9. Two objects for evaluating the performance of rope grasping on a spray container and a detergent container. The spray container has a bottle-neck, while the detergent container has a handle. The candidate rope caging grasps for these two objects are marked by the green curves.

samples, and the local frames $\mathbf{x}_p$, $\mathbf{y}_p$, and $\mathbf{z}_p$ at each sample using red, green, and blue colors. The middle image shows the computed caging grasps, where the green ones are the results returned by Algorithm 1 and the red ones are collided and deprecated candidates. The right image shows a clear view of the caging grasps, where the collided candidates are not plotted. During physical experiments, we use the "top" one or two candidates (the stretched rings that has largest difference from its initial states) of the rings shown in the left image to test rope caging and use the grasp candidate shown in the right figure to test soft robotic grippers. A robot will first approach the object with fully opened hand. Then, it closes the hand to form the planned cage. Finally, it fully wraps the object and immobilize it at a stable grasp.

V. Physical Experiment

In this section, we demonstrate the effectiveness of caging grasps based on minimal rings by using both a rope and a soft gripper. The grasps are tested on two containers as shown in Fig.9, which are served as the representatives for the objects with bottle-necks and for the objects with handles, respectively. The candidate caging grasps for these two objects computed by our approach are marked by the green curves in Fig.9.

A. Experiments with a Rope

In the first experiment, we use a rope to cage and grasp these two objects. As shown in Figs.10 and 11, the rope will start from a loose cage or at an unstable grasp. But after being stretched, it will eventually converge to a stable grasp as computed by our algorithm. For objects with more than one stable caging grasps, for instance the detergent container in Fig.9(b), the stretching direction will determine the final configuration of rope grasping.

B. Experiments with a Soft Gripper

In the second experiment, we use a soft gripper as shown in Fig.12. The soft gripper consists of two fingers made of silicone rubber and a driving mechanism. When the soft finger is dragged by a cable fixed at the fingertip and goes through the inside of the finger, it will bend the finger

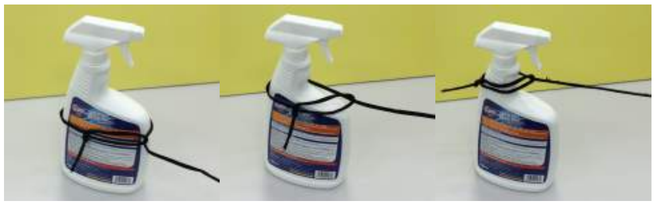

(a) (b) (c)

Fig. 10. Caging to grasping process (from left to right) for the spray container using a rope. The rope starts from a loose caging in (a), and then eventually reach a stable grasp in (c) after gradual stretching.

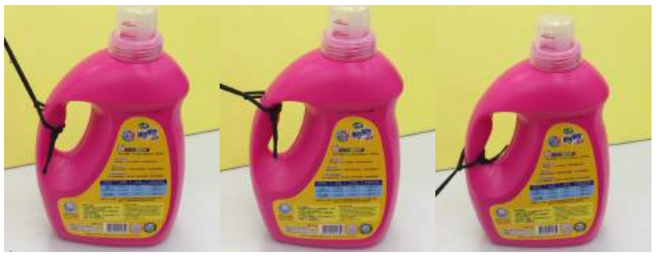

(a) (b) (c)

Fig. 11. Caging to grasping process for the detergent container using a rope. The rope starts at a non-stable caging in (a), and then will reach two different stable grasps as shown in (b) and (c), according to whether stretching the rope upwards or downwards.

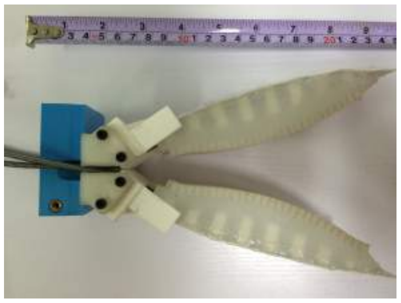


Fig. 12. A soft gripper that can achieve the caging grasp in a way similar to ropes.

inwards. When releasing the cable, the supporting board of the finger (the back side of the finger) will serve as a spring and resume the finger into its original position. The cable dragging the finger first goes through a spring tube, which is designed to guide the direction of the dragging force along the tube. The cable is connected to a linear motor, which can generate very large force for dragging the cable. With the help of position feedback mechanism, this mechanism can control the positions of fingers precisely.

The soft gripper is actually similar to a rope, though its shape cannot be arbitrarily changed but is non-linearly controlled by driving forces. However, we find the caging grasps computed by our approach still provides a good prediction about the final stable grasp for the test objects. As shown in Figs.13 and 14, the soft gripper starts from unstable grasps. After suitable shaking perturbation, the gripper will end up at grasping object with a stable pose. Since the soft gripper is not a perfect rope, the final stable grasp pose is

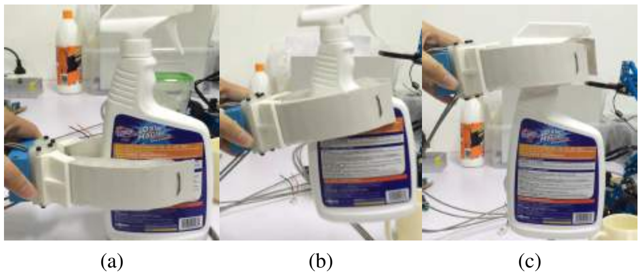


Fig. 13. Caging to grasping process (from left to right) for the spray container using a soft gripper. The grasping starts from a place as shown in (a) which is not stable and will slide upwards while shaking the gripper, and then eventually get immobilized at a stable grasp as (c).

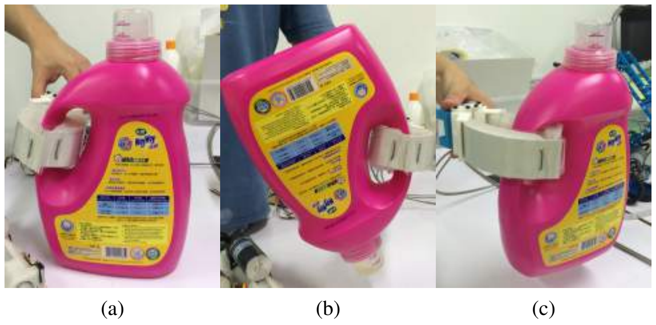


Fig. 14. Caging to grasping process for the detergent container using a soft gripper. The gripper starts to grasp at a place not stable as shown in (a) and will slide while shaking the gripper. Since the detergent has two isolated caging grasps, the soft hand will slide upwards or downwards according to the shaking direction, and eventually reach a stable grasp as (b) or (c).

not exactly same as the prediction provided by our algorithm, but it is close. In addition, similar to the rope case, we can find that for objects with more than one caging grasps, which grasp is achieved by the gripper is determined by the shaking direction and the pose of the object, as shown in Fig.14.

VI. Conclusion and Discussion

A new strategy called rope caging and grasping has been developed in this paper. Our method starts from generating an AGD field on the surface of a 3D model to be grasped. Critical points of the model's Reeb graph according to the AGD field are extracted to segment the model into topological branches. Initial rings are then constructed on all branches by the isocurves of the AGD field. To capture the local shape extremities on the given model, the initial rings are further stretched to become shorter and shorter in an iterative way. The resultant local minimal rings are employed as candidates of grasping to be further verified in the simulation. Physical experiments have been taken at the end of this paper to verify the performance of our approach.

The current implementation using C++ takes about 62 sec. to compute all the minimal rings for a model with 5k faces on a PC with i7-4700MQ 2.4GHz CPU. The long computation time is mainly caused by the unoptimized code for extracting initial rings. One of our future works is to speedup the computation in this step of initial ring generation. Another planned work is to test this rope caging and grasping work on an integrated platform with robotic arm and grippers so that both motion planning and grasping are considered and tested.

References

[1] A. Rodriguez, M. T. Mason , and S. Ferry, "From caging to grasping," *The International Journal of Robotics Research (IJRR)*, vol. 31, no. 7, pp. 886–900, June 2012.

[2] W. Wan, R. Fukui, M. Shimosaka, T. Sato, and Y. Kuniyoshi, "A new 'grasping by caging' solution by using eigen-shapes and space mapping," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 1566–1573.

[3] R. Diankov, S. Srinivasa, D. Ferguson , and J. Kuffner, "Manipulation planning with caging grasps," in *2008 IEEE International Conference on Humanoid Robots*, December 2008.

[4] T. K. Dey, J. Sun, and Y. Wang, "Approximating loops in a shortest homology basis from point data," in *Proceedings of the Twenty-sixth Annual Symposium on Computational Geometry*, 2010, pp. 166–175.

[5] J. Stork, F. Pokorny, and D. Kragic, "Integrated motion and clasp planning with virtual linking," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, Nov 2013, pp. 3007–3014.

[6] B. Calli, M. Wisse, and P. Jonker, "Grasping of unknown objects via curvature maximization using active vision," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, Sept 2011, pp. 995–1001.

[7] S. El-Khoury and A. Sahbani, "A new strategy combining empirical and analytical approaches for grasping unknown 3d objects," *Robotics and Autonomous Systems*, vol. 58, no. 5, pp. 497–507, 2010.

[8] T. Tsuji, S. Uto, K. Harada, R. Kurazume, T. Hasegawa, and K. Morooka, "Grasp planning for constricted parts of objects approximated with quadric surfaces," in *Intelligent Robots and Systems (IROS), 2014 IEEE/RSJ International Conference on*, 2014, pp. 2447–2453.

[9] D. Zarubin, F. Pokorny, M. Toussaint, and D. Kragic, "Caging complex objects with geodesic balls," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, Nov 2013, pp. 2999–3006.

[10] E. Zhang, K. Mischaikow, and G. Turk, "Feature-based surface parameterization and texture mapping," *ACM Trans. Graph.*, vol. 24, no. 1, pp. 1–27, 2005.

[11] J. Lin, X. Jin, Z. Fan, and C. Wang, "Automatic polycube-maps," in *Advances in Geometric Modeling and Processing*, 2008, pp. 3–16.

[12] X. Ni, M. Garland, and J. C. Hart, "Fair morse functions for extracting the topological structure of a surface mesh," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 613–622, 2004.

[13] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii, "Topology matching for fully automatic similarity estimation of 3d shapes," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, 2001, pp. 203–212.

[14] H. Hoppe, J. McDonald, T. Duchamp, T. DeRose, and W. Stuetzle, "Mesh optimization," in *Proceedings of ACM SIGGRAPH 1993*, January 1993, pp. 19–26.

[15] S. Izadi, R. A. Newcombe, D. Kim, O. Hilliges, D. Molyneaux, S. Hodges, P. Kohli, J. Shotton, A. J. Davison, and A. Fitzgibbon, "Kinectfusion: Real-time dynamic 3D surface reconstruction and interaction," in *ACM SIGGRAPH 2011 Talks*. ACM, 2011, pp. 23:1–23:1.

[16] C. C. L. Wang, "Cybertape: an interactive measurement tool on polyhedral surface," *Computers & Graphics*, vol. 28, no. 5, pp. 731–745, 2004.

[17] G. Reeb, "Sur les points singuliers d'une forme de pfaff complètement intégrable ou d'une fonction numérique [on the singular points of a complete integral pfaff form or of a numerical function]," *Comptes Rendus Acad. Sciences*, vol. 222, pp. 847–849, 1946.

[18] T. Kanai and H. Suzuki, "Approximate shortest path on a polyhedral surface and its applications," *Computer-Aided Design*, vol. 33, no. 11, pp. 801–811, 2001.

[19] V. Surazhsky, T. Surazhsky, D. Kirsanov, S. J. Gortler, and H. Hoppe, "Fast exact and approximate geodesics on meshes," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 553–560, 2005.

[20] Y. Liu, Z. Chen, and K. Tang, "Construction of iso-contours, bisectors, and voronoi diagrams on triangulated surfaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1502–1517, 2011.