Cooperative Grasping Through Topological Object Representation

Alejandro Marzinotto, Johannes A. Stork, Dimos V. Dimarogonas and Danica Kragic

***Abstract*— We present a cooperative grasping approach based on a topological representation of objects. Using point cloud data we extract loops on objects suitable for generating entanglement. We use the Gauss Linking Integral to derive controllers for multi-agent systems that generate hooking grasps on such loops while minimizing the entanglement between robots. The approach copes well with noisy point cloud data, it is computationally simple and robust. We demonstrate the method for performing object grasping and transportation, through a hooking maneuver, with two coordinated NAO robots.**

I. Introduction

Robots operating in domestic and industrial environments need to manipulate and transport objects which often fall outside their capabilities in terms of weight, shape or size. However, many of these objects can be handled by combining the efforts of multiple robots through coordinated multi-agent grasping and manipulation techniques. Our approach to cooperative multi-robot grasping builds upon our previous work presented in [1], [2] and relies on detecting loops in objects using a topological framework.

When multiple robots execute a task cooperatively, possible collisions with the environment *and* between the robots need to be handled. This problem is commonly addressed by approximating the robot's geometry using boxes, cylinders, spheres and other geometric primitives. Such shape elements are then fed to collision detection algorithms that indicate whether a pair-wise spatial overlap would occur or not. This procedure is computationally expensive, even for coarse approximations of the robot/object shape and has to be constantly repeated during planning and control.

We propose an alternative approach which is not intended to replace collision avoidance but rather to complement it. In short, we propose to leverage the same topological methods used to derive hook-grasping controllers, to keep the robots untangled. The contributions of this paper are:

1) Multi-agent topology-based grasping framework.
2) Collision avoidance solution through topology.

An overview of our system is shown in Fig. 1. The system builds upon our previous work on extracting loops to describe objects with holes [1] and a topological representation of the relationship between these loops and robots for grasping [2]. Our method proceeds as follows: First, we extract the object's cluster from a point cloud and perform Delaunay triangulation to obtain a simplicial complex that describes the object. Considering the object's geometry in a topological way, we extract loops from the simplicial complex and heuristically pick a target loop for entanglement by the robots. To create a grasp, we describe entanglements with a writhe matrix and control the robots' joints in a lower dimensional Eigen-space. In this process it is our goal to arrive at a *secure entanglement* condition which indicates that the loop is secured and the robots can proceed to transport the object.

The remainder of this paper is organized as follows: In Section II, we review related works in control and motion generation based on topological coordinates. Section III details loop extraction and theory for our topology-based representation. The grasping framework is presented in Section IV and Section V contains a real-world proof-of-concept demonstration using two NAO robots. In Section VI we give concluding remarks and elaborate on future research directions.

The authors are with the Computer Vision and Active Perception Lab., Centre for Autonomous Systems, School of Computer Science and Communication, KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden. e-mail: {almc|jastork|dimos|dani}@kth.se

II. Background and Related Work

Two fundamental concepts that we build upon are Gauss Linking Integral (GLI) and writhe matrix. [3] derives a computationally efficient way of calculating the entanglement of straight line segments, calculating the GLI without having to compute the integral directly. [4] synthesizes agent's motion with close contacts using topological coordinates such as writhe, center, and density. It allows us to avoid collisions for complex motions that require entanglement of body parts. The authors show how to modify the writhe matrix through rotation, translation and scaling transformations to obtain the desired entanglement.

The object collision avoidance is inspired by the ideas presented in [5]. The authors introduce a structure called interaction mesh that is used to represent spatial relationships between the agent and its environment. We generalize this mesh to represent spatial relationships between agents too. The minimization of the interaction mesh deformation yields motions that avoid inter-penetrations and — in our case — also collisions between agents. The authors use two types of energy in the minimization problem: deformation and acceleration; and they specify four constrains over the mesh: bone-length, positions, collisions and energy.

In [6] the authors redefine the energy functions of [5] to calculate the joint angles directly from the interaction mesh rather than having to extract them from the Cartesian coordinates. They include additional constraints that preserve the correctness of the motions generated. The scenario studied involves a NAO humanoid robot simulation for which not only the spatial relationship with the obstacle is considered,

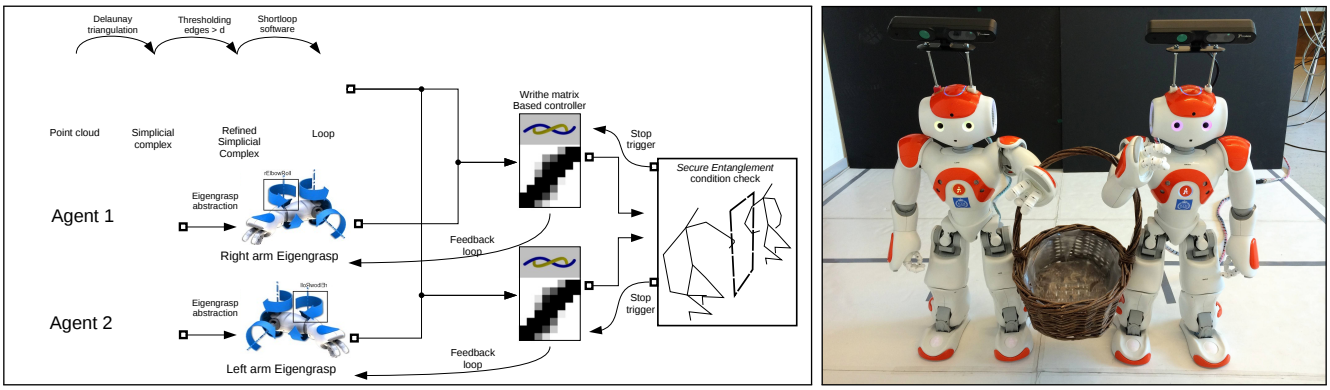


Fig. 1. *left)* Diagram of the topological control framework. *right)* Two NAOs carrying a basket cooperatively through the topological control framework.

but also how the weight distribution constrains the movement. To this end, the range of movement is restricted to lie within the support polygon of the feet.

To deal with the problem of body part inter-penetration, [7] introduces several techniques: RRT, PCA and IK, which are compared to the method of parameterizing body movements using topological coordinates. Traditionally, each posture would be encoded as a series of Euler angles. However, by calculating the GLI, the part of the data that interests us can be encoded by 3 numbers: writhe, center, and density. Both the agent and the object are defined using line segments that correspond to its skeleton disregarding the local geometry.

In order to execute interesting sequences of actions generated using topology based controllers, [8] simulates video-game characters using a Finite State Machine (FSM) that sits on top of the topology-generated actions. This creates a new level of abstraction because different actions can be scheduled in time depending on their outcome. In this work, we replace the concept of FSM with that of Behavior Trees (BTs) [9], [10] as they can represent the same types of action sequences with simpler notation. Additionally, BTs provide the benefit of being modular structures, which means they can be disaggregated into smaller parts or chained together to build more complex modules seamlessly.

Since we are interested in executing not only multi-agent grasps but also more complex sequences of actions using BTs, we investigated dynamic footstep planners which could be incorporated into our framework. For example, [11] proposes a method to parameterize a humanoid robot walking motion using the displacement of one foot with respect to the other. The authors test several planners such as A* and D* Lite, of which the latter is preferred because it does incremental searches. Such footstep planner has the precision required to drive a humanoid robot to the position where the loop is within arm's reach without colliding with it.

In [12], the authors present the idea of switching from a coarse motion walking function to a foot planner when the situation demands it. The foot planning parameterization is maintained with the improvement that now the search is performed using weighted A*. The implementation also incorporates ARA* which efficiently reuses previous information and progressively finds better solutions by decreasing the weights. The objects that clutter the environment are detected using a 3D sensor that is attached to the NAO's head. The planner has an adaptive level of detail which separates the region in two categories: traversable and non-traversable, determining the planner that ought to be used.

Regarding the problem of finding a feasible path to execute grasping, [13] uses RRT in combination with IK considering all the relevant constraints during the search. The robot is meant to interact with articulated objects such as drawers and doors while keeping its feet fixed. The authors compare the Jacobian and randomized methods for doing planning enforcing the benefits and limitations of each. The constraint manifold, which represents the limitations of robot self-collisions, is computed off-line. The robot stability is calculated by projecting the center of mass (CoM) to the ground plane and checking if it lies within the support polygon defined by the feet. Inspired by this work, we use a modified version of RRTs to execute the hooking grasp driven by topological coordinates in real-time.

In [14], the authors develop decentralized control strategies for groups of robots to move towards a goal position while maintaining a condition called *object closure*. Contrary to traditional force closure constraints, the condition used to transport objects in this framework is less stringent and therefore allows more tolerance to positioning errors. We take advantage of the same formalization to transport the object entangled with two agents in our demonstrator scenario.

Lastly, we investigated works on navigation in cluttered environments which could be adapted to work in conjunction with our framework. For example, [15] achieves almost real-time plans with bounded sub-optimality using a combination of a octree-based 3D world representation and an ARA* planner. It includes a review of several 3D collision checking algorithms that rely on partial subdivision of the space. The authors use three horizontal cuts of the PR2 robot shape at different heights to accelerate the collision avoidance by doing it in 2D first. If no collisions are detected at this stage, the algorithm calculates the full 3D collision test taking into account the geometry of the robot. The search algorithm ARA* relies on motion primitives which are concatenated to produce feasible paths. We consider this

work to be suitable for incorporation of entanglement notions that use the skeletal structure of the robot rather than arbitrary horizontal cuts to do a simplified object collision avoidance.

III. Preliminaries

A. Loop Extraction Algorithm

A point cloud obtained using a 3D sensor mounted on the robot is clustered and analyzed for holes: specifically, given the point cloud of the object's cluster, we obtain the Delaunay triangulation D which yields a simplicial complex $\mathcal{K}_{d_1}$ encompassing the object's volume, see [2] for details.

To detect holes in dense point clouds — where any two points that correspond to the solid part of the object are never apart more than d — we can first refine the simplicial complex by removing the edges which are larger than d. This yields a new simplicial complex $\mathcal{K}_{d_2}$, shown in Fig. 2, that approximates the shape of the object exposing the detectable holes [16] while keeping the relevant shape information.

Feeding this structure to the *ShortLoop* software [17], we obtain, in polynomial time, the shortest loops of the object. This corresponds to a set of piecewise linear loops/closed curves $S = \{l_1, l_2, \ldots, l_m\}$, which can be classified even further into graspable/non-graspable loops according to the size of the gripper, [2]. There exists other criteria to classify loops besides graspable/non-graspable, for example tunnel loops and handle loops, [2], [16]. In the current scenario, we assume that there is only one graspable loop on the object.

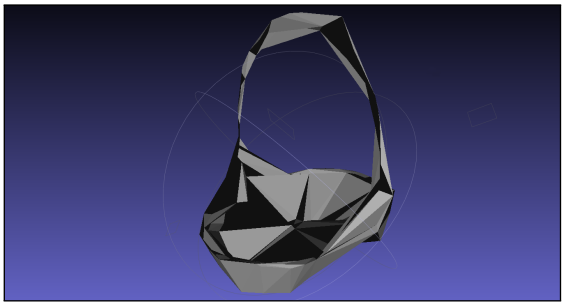


Fig. 2. Refined simplicial complex representation of the basket ($\mathcal{K}_{d_2}$) obtained from a point cloud by Delaunay triangulation and removing long edges.

B. Topological Representation

The topology space consists of coordinates that capture spatial relationships between objects and robots. There are three main topological coordinates: writhe, center and density. The *writhe* measures the degree in which two curves, γ_1 and γ_2, are entangled or twisted around each other. The writhe can be calculated through the Gauss Linking Integral (GLI) in continuous space using:

$$\mathrm{GLI}(\gamma_1, \gamma_2) = \frac{1}{4\pi} \iint_{\gamma_1 \gamma_2} \frac{\mathrm{d}\gamma_1 \times \mathrm{d}\gamma_2 \cdot (\gamma_1 - \gamma_2)}{\|\gamma_1 - \gamma_2\|^3} \qquad (1)$$

Clearly, there are many ways in which two curves can produce the same GLI value, and the election of the curve's label order (γ_1, γ_2) is arbitrary. To get additional information about the entanglement between two curves in topological space we define also the *center* and *density* of the twist, represented by c and d respectively.

The center c specifies the location of the main twist along the two strands (γ_1, γ_2). It is composed of two sub-coordinates (x_g, y_g) each of which is a scalar value and represents the relative location of the main twist along one of the two strands involved. The density d specifies the concentration of the twist along one strand with respect to the other. Evenly spread out entanglements yield lower (in absolute value) densities than non-evenly spread entanglements. Furthermore, d lies between $[-\frac{\pi}{4}, +\frac{\pi}{4}]$ and its sign tells us which strand is playing the major role in producing the entanglement.

We use topological coordinates to control a robot to reach a target topological configuration. This implies that we need a fast way of recomputing the topological coordinates, particularly the writhe matrix, so that we can use it as feedback to control the robot in real time. Borrowing the idea of approximating curves with multiple line segments from integral calculus, we can represent the internal structure of any robot and object with line segments. This hierarchical structure can be conveniently represented by a tree graph and is compliant with the standard definition of robots' joint specifications, i.e., URDF models.

Using this curve discretization on (γ_1, γ_2) we obtain (S_1, S_2) with n_1 and n_2 edges respectively. In this way, eq. (1) becomes a computation that can be done in parallel — leveraging multi-core GPUs — to obtain the entanglement between each line segment of S_1 and S_2. A convenient way of storing and representing this data structure is through what is called *writhe matrix* $T_{i,j}$ [3]. From such matrix, the calculation of the approximated GLI is straightforward using:

$$w = \mathrm{GLI}(S_1, S_2) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} T_{i,j} \approx \mathrm{GLI}(\gamma_1, \gamma_2) \qquad (2)$$

Maintaining the same notation, we can approximate the center $c = (x_g, y_g)$ of the entanglement for the discrete case with eq. (3). The center tells the location of the twist with respect to each strand, and it can be graphically inferred from the writhe matrix as shown in [4].

$$c = \Big(\frac{\sum_{i=1}^{n_1} \sum_{j=1}^{n_2} i\, T_{i,j}}{w} - \frac{n_2}{2}, \frac{\sum_{i=1}^{n_1} \sum_{j=1}^{n_2} j\, T_{i,j}}{w} - \frac{n_1}{2} \Big) \qquad (3)$$

Lastly, the density is defined as the angle between the main axis of the writhe matrix and the diagonals. Despite having a straightforward graphical definition, it is the most complex topological coordinate to calculate. We first note the fact that each strand could have a different number of edges, in this case the one with the largest number of edges must be filtered so that the writhe matrix has a square shape. This filtering process consists of grouping together the contents of several cells into a smaller number of them so that the number of rows and columns is the same.

At this stage, we must first filter the writhe matrix using an 'edge to edge' threshold. The filter outputs 1 if $T_{i,j} \geq T_d$ or

0 if $T_{i,j} < T_d$. Using the discrete 2D indices of each of the points whose value is 1, we can formulate a PCA problem that allows us to approximate the value of the direction along which these points are concentrated. The eigenvector with the highest eigenvalue V_d gives us the direction of the highest variance, while the eigenvalue itself tells us how much is the data concentrated in this direction. We find the angles between V_d and the diagonals of the matrix, which choosing to bottom-left corner of the matrix as the reference frame for the PCA analysis, would correspond to the vectors [1,1] and [1,-1]. Lastly, we obtain the density from these two angles by choosing the one with the lowest absolute value and placing the sign corresponding to the quadrant where the vector lies, i.e., 'plus' for *north* / *south*, and 'minus' for *east* / *west*.

The insight used in [16] was that if we have 2 non-intersecting closed curves for which the $\mathrm{GLI} \neq 0$, then they cannot be "pulled apart". For example, consider 2 such loops representing the gripper of a robot and the handle of an object; if the GLI between them is not zero, they cannot be unchained without breaking either of them. This situation when the object cannot be "pulled apart" from the robot(s) is what we call from this point onwards *secure entanglement*.

In this paper, we elaborate more general conditions that can be used to guarantee secure entanglements when considering two extensions. First, we broaden the idea of robot grasping to encompass not only gripper/object interactions, but also whole body/object interactions. Second, we consider that grasps can be executed cooperatively by more than one agent at a time on the same object and hole. For the first extension we show that secure entanglements can be achieved using, for example, the arms of the robot when the width of the loop is too large for the gripper. For the second extension we show that secure entanglements can be achieved by a group of robots even when none of them is capable of maintaining a secure entanglement on its own.

1) Robot Representation: In our previous work [16], it was required to identify at least one pair of opposable fingers for generating grasps. In the present paper, we propose a more general framework that allows to generate caging grasps that can be executed by both hands and arms. Our algorithm is concerned with finding a *secure entanglement* using the set of available joints.

2) Object Representation: The *ShortLoop* algorithm returns a set of points in 3D which define the coordinates through which the loop passes. Assuming the loop is approximately planar we can use the loop coordinates to formulate a minimization problem to fit a plane to the loop. Such plane — for relatively flat loops — approximates the loop's normal vector and we use it to define heuristics for the robot's approach direction. When several robots are to entangle with the same loop, it becomes necessary to introduce a notion of *loop effective traversable area* and how to distribute it. To do this, we take the center of the loop and draw the connecting lines to the loop's vertices. The set of triangles that result are used for calculating and distributing the area through which each agent should entangle the loop.

Naturally, the limitations of the platform will affect the choice of heuristics for assigning the entanglement to each agent. For example, when using two NAOs to grasp the basket, it is natural to split the loop 'vertically', whereas when doing the same thing with two Youbots, it may actually be just as feasible to divide the loop 'horizontally'. In either case, we assume that the position from which the robots will begin the entanglement maneuver allows the robot to reach the object without moving their feet or base.

The object description presented in [1] was given by $(S, \mathcal{K}_d, \{G_s | s \in S\})$ where S are the discretized shortest loops, $\mathcal{K}_d$ is the simplicial complex, and G_s are clasping targets. In this work we replace the notion of clasping targets G_s by the notion of *weighted entanglement analysis*, and we extend the object description to incorporate knowledge of the agents supposed to grasp it. This means that we have a function $\mathcal{F} : E \to A$ which maps each object edge E to the agent A that is going to entangle with it.

IV. Topological Grasping Framework

A. Topology Driven Controller

The relationship between differential changes in the robots' joint space and differential changes in its writhe matrix is complex to obtain even for simple robot embodiments and objects. We take a different approach than [4] to simplify the computation. We start the robot at the initial pose from which we know the object is potentially hookable without moving the feet. Formally, we have a function $Z : A \times J \to \{1, 0\}$ that for each agent A specifies the joints J that the controller is allowed to modify. For example Z will take the value zero for joints related to the legs of the NAO and the value one for joints related to the arms and other entangling body parts.

The aim is to explore the state space of the robot, particularly that of the joints intended to perform the hooking. We use a variation of the RRT algorithm that leverages the knowledge of the quality of each node in the search. This yields better results because we can quantify and therefore expand nodes that score higher, rather than simply expand nodes that belong to the non-collision state space.

For each valid random pose sampled we calculate the writhe matrix between the robot and the loop, plus the writhe matrix between the robot and any other agent present. The agent/object matrix is used to drive the robot towards high values of entanglement with the loop, whereas the other agent/agent matrices are used to drive the robot towards low values of entanglement with the obstacles (other robots). Clearly, this maximization problem is too complex to be handled all at once and for all joints simultaneously. For this reason we take a different approach achieving real-time execution by carrying out a *weighted entanglement analysis*. We first note that the number of edges in the robot skeleton, unlike the number of edges in the loop, is constant. This allows us to reward with higher weights the edges in the robot whose entanglement with the object is more important.

The scoring function, which is described in Sec. IV-B, must not only reward high entanglements, but also prevent the collision of the robot with the loop. To this end we set a

threshold on the individual value that the elements of $T_{i,j}$ can take. The reason behind this is that the definition of the GLI as described by eq. (1) has the distance between γ_1 and γ_2 in the denominator. This means that, as the edges of the loop and the robot get closer, the value of the corresponding $T_{i,j}$ gets larger. By penalizing high $\|T_{i,j}\|$ we encourage $\text{GLI} = \sum_{i,j} T_{i,j}$ to grow while avoiding collisions.

In our previous work [1], the clasping target on the loop γ was a point $p \in \gamma$ and its tangent $\dot{p}$. In the present work, this concept was replaced with a random search that drives the robot towards entanglement values that fulfill certain properties. We acknowledge that inappropriate thresholds for $T_{i,j}$ (too high or too low) lead to poor entanglements or collisions respectively. For the situation where the topological approach fails to prevent collisions, it is possible to apply another collision avoidance algorithm using $\mathcal{K}_{d_2}$ as a collision model for the object.

B. RRT Modified Algorithm Overview

In RRTs, a region of the search space is labeled as goal and it indicates the end of the algorithm. In our case, this goal region is not static: the condition that guarantees the loop cannot escape depends on more than one robot and it will vary with time as they move. For this reason, we evaluate the goal dynamically using a boolean function that we call *secure entanglement*, which depends on all the robots involved.

The main reason behind introducing the modified RRT algorithm is to achieve real-time performance for groups of agents. This means that we need to perform search using the information that we gain through expanding and evaluating nodes. A key concept in the modified RRT is the scoring function that indicates the nodes that ought to be expanded first because they are 'better'. The particular scoring function that we use is called weighted entanglement and it is represented by eq. (4). Intuitively, the first term of the summation rewards high entanglement values of $T_{i,j}^{a_m \to o}$ for the agent m/object writhe matrix, the second term of the summation penalizes high entanglement values of $T_{i,j}^{a_m \to a_n}$ for the agent m/agent n writhe matrix (or matrices if there are more than 2 agents), and the third term of the summation encourages a uniform growth of the entanglement by penalizing each $T_{i,j}^{a_m \to o}$ that increases disproportionally from the average $\bar{T}^{a_m \to o} = (n_1 n_2)^{-1} \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} T_{i,j}^{a_m \to o}$.

$$\text{score}(S_1, S_2) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \Big(\alpha_{i,j} T_{i,j}^{a_m \to o} - \beta_{i,j} T_{i,j}^{a_m \to a_n} - \zeta_{i,j} \|T_{i,j}^{a_m \to o} - \bar{T}^{a_m \to o}\|^2 \Big) \quad (4)$$

We use a goal-less entanglement search algorithm because it can exit when a certain condition is satisfied over the set of agents. The scoring function defines exactly what it means to improve the entanglement and the secure hooking condition is the trigger that stops the search expansion for all the agents. We note that there exists an infinite number of joint configurations that lead to a *secure entanglement* but our algorithm will only find one (the first joint configuration that the random search yields for that particular run).

C. Eigen-Entanglement

The approach to eigen-entaglement is inspired by the idea of eigen-grasps [18] where PCA is used to determine the parameters causing the largest variance in a grasp database. We thus generalize the concept of eigen-grasps to also encompass entanglements. This allows us to reduce the dimensionality of the search so that instead of exploring the whole joint space, we do so in a restricted eigen space that covers more than 80% of the entanglements.

D. Escaping Distance

Given that we aim at generating caging grasps that do not fixate the object, robot motion may cause changes in the object's pose after it has been grasped. To simplify the problem we limit the scope of this section referring only to situations where the object is securely hooked using exactly one agent. Under this assumption, we look into how the object moves under internal or external forces for the worst case scenario. This allows us to estimate a virtual convex surface around the part of the robot hooking the object. We use this surface for collision checking and as a safety distance that guarantees the object will not collide with anything if it remains hooked.

To obtain this virtual surface we divide the problem into two phases: first, we disregard the shape of the gripper and calculate how far the object could move if it was hooked in a way such that the "abstract gripper" was in contact with the loop; second, we incorporate the knowledge of the gripper's shape so that we know exactly how far from the robot could the object move if it happened to slide. We say gripper in this context referring to the part of the robot caging the object.

Consider the simplicial complex representation $\mathcal{K}_{d_2}$ described earlier and assume that the *ShortLoop* [17] algorithm found a loop that runs through a set of vertices V in the simplicial complex, such that $Q = \mathcal{K}_{d_2} \backslash V$ are the vertices of $\mathcal{K}_{d_2}$ which do not form part of the loop but are in $\mathcal{K}_{d_2}$. To solve the first phase we proceed as follows: 1) Hypothesize that the gripper is in contact with one of the vertices v of the loop. 2) For this vertex, loop through all the vertices $q \in Q$ building a matrix $\mathbf{D}$ of Euclidean distances(v, q). 3) Repeat the above procedure for all $v \in V$ adding corresponding rows to the matrix. 4) Take the maximum of all the entries in the matrix to obtain the resulting distance r_{ext}. The procedure is summarized in Alg. 1.

Algorithm 1: Escaping Distance - Phase 1

```
for v ∈ V do
    for q ∈ Q do
        D ← dist_Euclidean(v, q)
    end
end
r_ext ← max(max(D))
```

To solve the second phase we proceed as follows: 1) For every vertex $g \in G$, where G is the set of vertices that describe the shape of the gripper, we create a spherical collision structure centered in g and with radius r_{ext}. 2) We loop through these collision structures appending them to each other as with a 'union' operation. In this way, we can perform feedback control by measuring (indirectly) how far could the object slip, and then tightening or loosening the gripper to decrease or increase the mobility that we allow for the hooked object. The procedure is summarized in Alg. 2.

For situations where multiple agents are hooking the same loop, the problem is more complicated because we could have a finite collision shape without necessarily having either of the robots chained to the loop. This happens when the set formed by the agents A and the loop S satisfies what we call the *secure entanglement/hooking condition*.

Algorithm 2: Escaping Distance - Phase 2

```
collision ← ∅
for g ∈ G do
    collision ← collision ∪ sphere(r_ext)
end
```

E. Secure Entanglement

This condition is defined for idealized loops and robots, i.e., those composed of mathematical lines rather than volumetric shapes. This implies that the secure hook estimated through the idealized representation overestimates the real condition that takes into account the volume and geometry. The problem that we are dealing with consists of finding — for a closed loop and a set of manipulators — whether the loop can be separated arbitrarily far away from the manipulators without bending or breaking it. For the case where we have only one manipulator and one loop, the secure hooking condition is true when the: $\text{GLI}(G, S) \neq 0$ and the manipulator edges themselves form a closed loop. When two agents are entangled with the object, it is no longer necessary that either of them has a secure entanglement for the system to fulfill such condition. This means that we could have a system where two robots do not have secure entanglement on their own, but as a whole they do prevent the loop from escaping arbitrarily far away from them.

If either of the robots happens to have secure entanglement on its own, the two agent system also satisfies the condition. Otherwise, we proceed as follows: 1) We assume that each robot on its own is *virtually linked* to the object [1], i.e., when completing the missing edge to close the manipulator it turns out that $\text{GLI}(G, S) \neq 0$, and that their edges define convex shapes. 2) Similar to the procedure that we used to calculate r_{ext}, we calculate r_{int} as the maximum element of the matrix that stores the Euclidean distance between every pair of loop vertices $v \in V$, i.e., we idealize the loop as a circle with radius equal to the largest vertex to vertex Euclidean distance appearing in the loop. 3) For each manipulator, we label its vertices as belonging to the extremes (*ext*) or not (*int*). 4) For each agent $a \in A$ we loop through the internal vertices v_{int} of its manipulator calculating the $d_{\text{prel}} = \text{dist}_{\text{Euclidean}}(v_{\text{int}}^{\text{agent}=a}, v_{\text{ext}}^{\text{agent}\neq a})$, if $r_{\text{int}} < d_{\text{prel}}$ for both robots, and all combinations of internal and external vertices, we say the object is preliminarily hooked. 5) We draw virtual links that join the external nodes of one robot with the other such that these new edges do not cross. 6) We check if the area defined by this new virtually closed manipulator is traversed by the loop or not, if it is not we have that the individual manipulators are facing away from each other and we can finally say that the loop is securely entangled, otherwise we have that one or both of the manipulators are facing inwards meaning that they are not actually keeping a secure hook even if they preliminarily appear to do so. The procedure is summarized in Alg. 3.

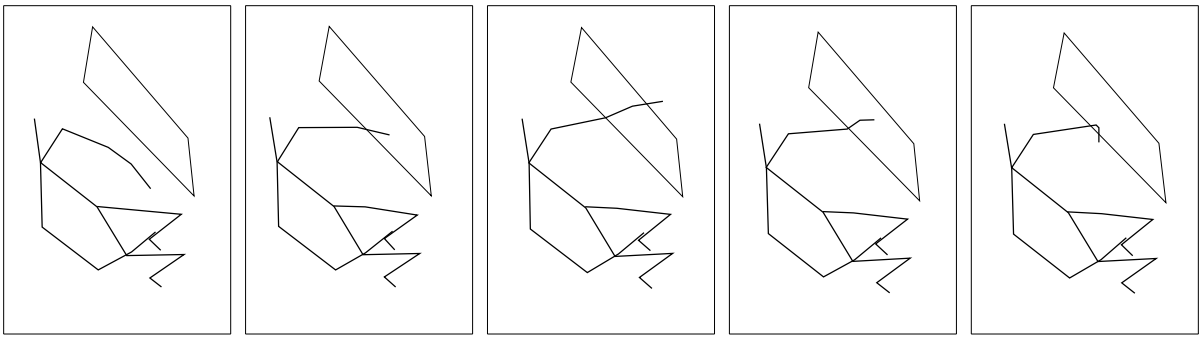
Algorithm 3: Secure Entanglement

```
for v_1 ∈ V do
    for v_2 ∈ V do
        D ← dist_Euclidean(v_1, v_2)
    end
end
r_int ← max(max(D))
pre_hook ← True
for a ∈ A do
    for v_int ∈ V_int do
        d_prel ← dist_Euclidean(v_int^{agent=a}, v_ext^{agent≠a})
        if not r_int < d_prel then
            prel_hook ← False
            break
        end
    end
end
v_manip ← virtually_link(a_1, a_2)
if GLI(v_manip, loop) = 0 then
    secure_entanglement ← True
else
    secure_entanglement ← False
end
```

For situations where $n > 2$ robots are involved, we could analyze if there is a secure entanglement between any two robots. If such check is positive, we know the object is securely hooked. Otherwise, it could still happen that the loop is securely hooked due to the interaction of three or more agents simultaneously. Deriving a *necessary and sufficient* condition that guarantees secure hooking between arbitrary number of agents and a loop has been left for future work.

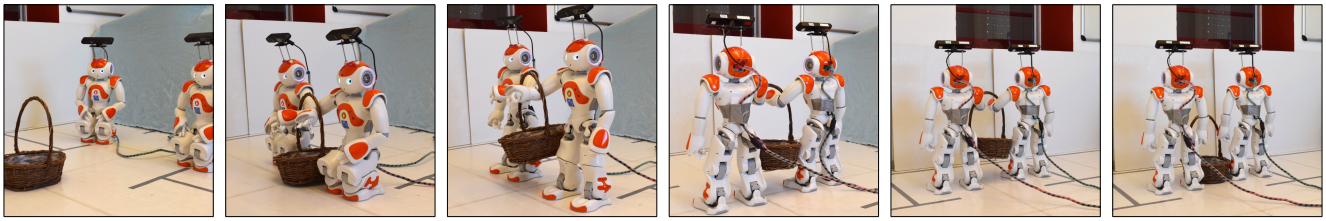
V. System Demonstration

The system that demonstrates the proposed topological multi-robot grasping framework consists of two NAO humanoid robots equipped with 3D sensors, a local positioning system (which for simplicity was replaced with robot odometry), and the object with a graspable loop which in this case is a basket. In the following subsections we describe the task used to demonstrate the topological grasping framework and how each of these parts are coupled together through ROS.



(a) GLI $= -0.2065$ (b) GLI $= +0.1113$ (c) GLI $= +0.2765$ (d) GLI $= +0.4515$ (e) GLI $= +0.6109$

Fig. 3. Selection of five nodes evaluated by our RRT modified algorithm. The NAO skeleton entangles a simplified loop with four edges using its left arm while keeping the rest of the body still. Notice the increasing entanglement value achieved by exploring more nodes.



(a) Scanning Basket (b) Entangling Basket (c) Lifting Basket (d) Turning Basket (e) Transporting Basket (f) Releasing Basket

Fig. 4. Basket *secure entanglement* with transportation, (basket & robots initial positions known, odometry used for navigation). This figure is merely for illustrative purposes, for the real run of the system refer to the YouTube video: "Cooperative Grasping Through Topological Object Representation".

A. Task Description

The evaluation scenario is as follows: 1) two NAOs extract a graspable loop from the point cloud of a basket while in crouching position, 2) the NAOs walk towards the basket until they reach a position from which they can grasp it, 3) the NAOs move their arms maximizing the writhe matrix until the secure entanglement condition is satisfied. 4) the NAOs stand while keeping the same arm pose and walk cooperatively carrying the basket to deliver it to the goal.

The reason the NAOs must perform the loop recognition before walking to the place where the grasping will be performed is because the 3D cameras that we use have a distance limitation. This means that the NAO is unable to detect the loop when it is close enough to reach the basket, or equivalently that it is unable to reach the basket when standing far enough so that the point cloud is complete. In a practical sense, this is translated in the fact that we have to use odometry to recalculate where would the basket be after we have moved the NAO from the observing pose to the grasping pose. The scenario described is represented graphically in Fig. 4.

B. Behavior Tree Middle Layer Representation

The scenario described above requires the switching between several controllers as certain conditions are achieved, e.g. detect the loop, move to grasping pose, entangle, transport and release basket. This scheduling is achieved using BTs [9], [10] instead of the traditional FSMs.

Behavior Trees can handle fallback conditions and have plenty of interesting features. However, for the current task only the basic functionality of BTs is required, i.e., the sequence node. Informally speaking the sequence node (represented by $\rightarrow$) executes its children from left to right one after the other for as long as they continue to succeed. One BT consists of an interconnection of action nodes (controllers) and control-flow nodes (scheduling the controllers in time according to their outputs). Each agent runs one BT which deterministically specifies what the agent should be doing at each time step. Synchronization decorators[1] can be used to have the agents executing certain actions simultaneously.

C. RRT Modified Search

To drive the robot towards high entanglement values we have used eq. 4 in conjunction with a modified RRT search. This search expands nodes in the robot's joint state space using eq. 4 as a quality measure that indicates the nodes that ought to be expanded first because they yield 'better' entanglement values between the robot and the loop.

To simplify the understanding of the pictures we have restricted the scenario to consider only one NAO. The robot is meant to entangle a simplified loop which consists of four edges as shown in Fig. 3. The NAO starts from a crouching position and we enable our algorithm to control only the joints that correspond to the left arm.

As we previously mentioned, the triggering condition that stops the search is what we call the *secure entanglement*. However, in the example represented in Fig. 3 the NAO by itself cannot reach such condition because we only allow it

[1]For details regarding the functionality of BTs and our open source implementation[2] refer to [9].

to move one arm. This means that we have to exit the search when the GLI does not increase substantially for some time.

In the scenario where two NAOs perform the same maneuver, they exit when the *secure entanglement* is achieved. At this point they stand up simultaneously, preserving the distances between their parts. This guarantees that the object which is securely entangled will not escape. Unfortunately, the transportation phase does not have this certainty because the NAOs bounce sideways and drift away while walking.

VI. Conclusions

We have presented an approach to topology driven multi-robot grasping and hooking. Our approach relies on global topological features [2], inferred from noisy point cloud data. Even though not explicitly shown, our method is suitable for deformable/non-rigid object grasping as long as the properties of the topological space are maintained, i.e., existing holes are preserved.

Caging and entanglement-based grasps such as the ones that can be achieved through the methods proposed in this paper are important because they allow more flexibility and error tolerance than force-closure grasps. We humans employ them on a daily basis when opening doors, holding hand rails, carrying bags, manipulating tools, etc.

Clearly, not all the objects that we wish to cage or entangle with are easily recognizable using the computer vision algorithms that exists today. For this reason, more work needs to be done in the area of perceiving, understanding and classifying loops in order to unlock the full potential and applicability of the methods proposed in this paper.

In future work we plan to address more in detail the problem of obstacle avoidance using topologically inspired methods. This will permit us to make a reliable comparison with the state of the art in collision avoidance and evaluate the true potential of topology to deal with this problem. Lastly, we are interested in deriving the necessary and sufficient condition that guarantees secure entanglement for groups of an arbitrary number of robots and a loop.

Acknowledgment

This work has been supported by the Swedish Research Council (VR) and the European Union Project RECONFIG, `www.reconfig.eu` (FP7-ICT-2011-9, Project Number: 600825) and Project FLEXBOT (FP7-ERC-2011-StG, Project Number 279933) The authors gratefully acknowledge the support.

We also thank: *Michele Colledanchise* for the ideas and useful comments provided for this paper.

References

[1] J. A. Stork, F. T. Pokorny, and D. Kragic, "Integrated Motion and Clasp Planning with Virtual Linking," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'2013)*, Tokyo, Japan, 2013.

[2] J. A. Stork, F. T. Pokorny, and D. Kragic "A Topology-based Object Representation for Clasping, Latching and Hooking," in *IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS'13)*, Atlanta, USA, 2013.

[3] K. Klenin and J. Langowski, "Computation of Writhe in Modeling of Supercoiled DNA," *PubMed Biopolymers*, April 2000.

[4] E. S. L. Ho and T. Komura, "Character Motion Synthesis by Topology Coordinates," in *Computer Graphics Forum (Proc. Eurographics 2009)*, P. Dutr'e and M. Stamminger, Eds., vol. 28, no. 2, Munich, Germany, Mar. 2009.

[5] E. S. L. Ho, T. Komura, and C.-L. Tai, "Spatial Relationship Preserving Character Motion Adaptation," *ACM Transactions on Graphics*, vol. 29, no. 4, pp. 1–8, 2010.

[6] E. S. L. Ho and H. P. H. Shum, "Motion Adaptation for Humanoid Robots in Constrained Environments," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, may 2013, pp. 1–6.

[7] E. S. L. Ho, H. P. H. Shum, Y.-m. Cheung, and P. C. Yuen, "Topology Aware Data-Driven Inverse Kinematics," vol. 32, no. 7, Oct. 2013.

[8] E. S. L. Ho and T. Komura, "A Finite State Machine Based on Topology Coordinates for Wrestling Games," *Comput. Animat. Virtual Worlds*, vol. 22, pp. 435–443, Sep. 2011.

[9] A. Marzinotto, M. Colledanchise, C. Smith, and P. Ögren, "Towards a Unified Behavior Trees Framework for Robot Control," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, June 2014.

[10] M. Colledanchise, A. Marzinotto, and P. Ögren, "Performance Analysis of Stochastic Behavior Trees," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, June 2014.

[11] J. Garimort and A. Hornung, "Humanoid Navigation with Dynamic Footstep Plans," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, May 2011, pp. 3982–3987.

[12] A. Hornung, D. Maier, and M. Bennewitz, "Search-Based Footstep Planning," in *Proc. of the ICRA Workshop on Progress and Open Problems in Motion Planning and Navigation for Humanoids*, Karlsruhe, Germany, May 2013.

[13] F. Burget, A. Hornung, and M. Bennewitz, "Whole-Body Motion Planning for Manipulation of Articulated Objects," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, May 2013.

[14] G. A. S. Pereira, M. F. M. Campos, and V. Kumar, "Decentralized Algorithms for Multi-Robot Manipulation via Caging," *I. J. Robotic Res.*, vol. 23, no. 7-8, pp. 783–795, 2004.

[15] A. Hornung, M. Phillips, E. G. Jones, M. Bennewitz, M. Likhachev, and S. Chitta, "Navigation in Three-Dimensional Cluttered Environments for Mobile Manipulation," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, St. Paul, MN, USA, May 2012.

[16] F. T. Pokorny, J. A. Stork, and D. Kragic, "Grasping Objects with Holes: A Topological Approach," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'13)*, Karlsruhe, Germany, 2013.

[17] O. Busaryev, T. Dey, J. Sun, and Y. Wang, "ShortLoop Software for Computing Loops in a Shortest Homology Basis," Software, 2010.

[18] M. T. Ciocarlie, "Low-Dimensional Robotic Grasping: Eigengrasp Subspaces and Optimized Underactuation," Ph.D. dissertation, Columbia, 2010.