# Persistent Homology for Effective Non-Prehensile Manipulation

Ewerton R. Vieira, Daniel Nakhimovich, Kai Gao, Rui Wang, Jingjin Yu and Kostas E. Bekris

***Abstract*— This work explores the use of topological tools for achieving effective non-prehensile manipulation in cluttered, constrained workspaces. In particular, it proposes the use of persistent homology as a guiding principle in identifying the appropriate non-prehensile actions, such as pushing, to clean a cluttered space with a robotic arm so as to allow the retrieval of a target object. Persistent homology enables the automatic identification of connected components of blocking objects in the space without the need for manual input or tuning of parameters. The proposed algorithm uses this information to push groups of cylindrical objects together and aims to minimize the number of pushing actions needed to reach to the target. Simulated experiments in a physics engine using a model of the Baxter robot show that the proposed topology-driven solution is achieving significantly higher success rate in solving such constrained problems relatively to state-of-the-art alternatives from the literature. It manages to keep the number of pushing actions low, is computationally efficient and the resulting decisions and motion appear natural for effectively solving such tasks.**

## I. Introduction

In order to retrieve a target object from clutter, a robotic arm may first need to relocate other objects that are blocking access. Such a task appears often in applications ranging from service robotics (e.g., taking out a can of soda from the fridge) to logistics (e.g., retrieving an ordered product from the shelf of a grocery store). In many cases, especially in homes, the workspace is cluttered and unstructured. But it can still allow a combination of planar, non-prehensile pushes to clear blocking objects and prehensile grasping to pick the target object. This paper focuses on making such strategies for object retrieval more efficient so as to equip robot assistants with this useful skill.

While there are methods for realizing this robotic skill [1]–[3], human-level performance in terms of efficiency and smooth, natural motion has yet to be achieved. When humans perform such tasks, they often perform an implicit "object grouping" so as simultaneously push multiple objects and find the least number of pushes before the target can be retrieved. Building on that observation, this work, explores topological tools to systematically identify how objects can be grouped into manageable clusters (connected components in topological terms). Once the objects are grouped the approach identifies pushing actions that are effective in clearing the blocking objects. In particular, persistent homology (PH) provides a persistence diagram for selecting clusters

The authors are with the Department of Computer Science and DIMACS (the Center for Discrete Mathematics and Theoretical Computer Science), Rutgers University, NJ, USA. Email: {rw485, kg627, dn332, jy512, kb572}@cs.rutgers.edu and er691@rutgers.edu. The work is supported in part by an NSF HDR TRIPODS award 1934924.

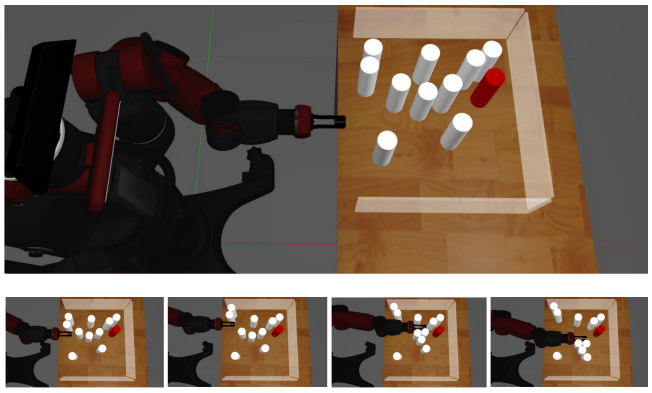

Fig. 1. (top) An example setup where a model of the Baxter robot and a cluttered set of cylindrical objects are modeled in a physics engine. The target object is highlighted and access to it is blocked by the white objects. (bottom) A solution sequence computed by the proposed topology-driven method. The arm rearranges the blocking objects in order to reach its target.

(connected components), in order to group objects that can be pushed together in one action.

The dynamics of such pushes are rather complex. A robot arm is usually not designed with push actions in mind. Such operations are in general underactuated, non-prehensile primitives that complicate the achievement of desirable end positions for the objects. They involve complicated dynamics as objects may collide each other. This work does not aim to understand or model the dynamics of such pushing operations over groups of objects. It shows, however, that reasoning about the topology of the configuration space is still helpful in this context and allows clustering objects that can be relocated effectively under non-prehensile manipulation. In other words, clusters of the objects given by PH can be more reliably pushed together and result in fewer pushes to solve such problems.

Two methods are presented here for selecting connected components of blocking objects in such setups. (A) For each configuration instance, the PH Informed Actions approach described in Algorithm 4 selects the closest connected component to the arm, that is minimal, which persists in the persistence diagram (taking into account the restrictions imposed by the object and arm size). (B) The PH Informed Search approach described in Algorithm 5 considers all connected components that persist for different radii in the persistence diagram. It builds a tree with nodes, each representing a configuration of the space, and propagates by performing push actions until it gets success or failure. Finally, it searches for the path that leads to success and minimizes the time for performing the pushing actions.

To the best of the authors' knowledge, this is the first application of these topological tools in this domain. Simu-

lated experiments using a physics engine and a model of the Baxter robot show that the proposed topology-driven solution is achieving significantly higher success rate in solving cluttered problems in constrained, shelf-like workspaces. The comparison points include randomized baselines as well as alternatives from the literature. The proposed strategy manages to keep the number of pushing actions low, executes effective pushing operations given the physical modeling of the engine, is computationally efficient, and produces motions which appear natural for effectively solving such tasks.

## II. Related Work

Many manipulation tasks, such as pick-and-place operations and object rearrangement , can be solved by using **prehensile actions**, where the robot grasps one object at a time. With relatively predictable movement of the objects, the objective is primarily focused on minimizing the number of grasps to fulfill the task [4]–[6] , or maximizing the number of objects to pick within a given time limit [7] . Prehensile manipulation, however, may require good knowledge of the objects' 3D shape or pose, which can be challenging in cluttered setups.

For manipulation setups where it is difficult or slow to perform grasping, **non-prehensile actions** are used for reconfiguring multiple objects at a time, which enables large-scale object manipulation [8], [9]. Pushing actions are preferred in tasks, such as bin picking and sorting [10]–[12], as they can be performed with smaller and simpler end-effectors that can easily fit in a cluttered, constrained space. In harder problems, pushing and grasping actions are used interchangeably throughout the task [13], [14]. As the effect of pushing is less predictable, there are efforts that focus on better estimating the outcome of pushing actions [15]–[17]. This paper utilizes pushing actions and performs topological reasoning to identify a group of objects, which can be pushed simultaneously so as to clear the path for approaching a target object to be retrieved.

Prior efforts in **object retrieval** under clutter have focused on identifying which objects to relocate so as to enable a collision-free path to reach the target object [18], [19]. To improve the success rate of such tasks, human interaction has been considered to supply a high-level plan, which informs an ordered sequence of objects and approximate goal positions [3]. A fast kinodynamic planner that takes advantage of dynamic, non-prehensile actions has been proposed without the assumption of quasi-static interactions [20]. The method proposed in this paper utilizes persistent homology to improve the efficiency of pushing, which outperforms some of these prior efforts [3], [20] in terms of the number of pushing actions and planning time. A similar problem relates to pushing a target to the desired goal position [21], which has been approached via learning an optimal policy from visual input [22], computing effective manipulation states and actions [23], or a value function based heuristic [2].

Topological reasoning has been more frequently applied to robotics applications with high-dimensionality of configuration space for high degree of freedom (DoF) manipulators. Constraint manifolds are introduced to plan feasible paths in configuration spaces with multiple constraints [24]–[26]. Topological techniques can also be deployed to verify path non-existences [27], [28]. Persistent homology is used to classify a class of trajectories with varying task-specific properties such as clearance of obstacles over the largest range of thresholds [29] or path-connectedness [30]. Here, the work utilizes persistent homology to automatically identify connected components of blocking objects so as to maximize the pushing efficiency.

## III. Problem Setup and Notation

Let $\mathcal{W} \subset \mathbb{R}^2$ be a bounded polygonal region, where a set of uniformly-shaped cylindrical *movable obstacles* $\mathcal{O} = \{o_1, \ldots, o_n\}$ and a similarly shaped target object $\mathcal{T}$ reside. A robotic arm is assumed to be able to reach with its gripper $g$ objects at any position in $\mathcal{W}$. The arm is not able to pick objects with overhand grasps or lift them due to limited accessibility. The robot arm may access the interior of $\mathcal{W}$ from one edge of $\partial\mathcal{W}$. Collisions between the arm and the boundary of $\mathcal{W}$ beyond that edge are not allowed. In other words, the robot arm moves along a 2D plane where the objects are located while respecting the geometric constraints of the workspace. The robot geometry assumed by the methods corresponds to a gripper $g$ attached to a cylindrical link representing the rest of the arm's geometry. Given the gripper's pose $s$, the location of the link attached to the gripper is fully specified. For the arm to be able to reach the target $\mathcal{T}$, the gripper must be able to grasp $\mathcal{T}$ and the cylindrical link of the arm must be collision free with all of the movable obstacles and the workspace boundary. In the accompanying experiments, the robot is actually a 7-dim. articulated Baxter robotic arm. The part of the arm inside the workspace is always contained by a cylindrical approximation considered by the proposed methods. The objective is for the robotic arm to reach and grasp the target $\mathcal{T}$, which may require moving some of the obstacles in $\mathcal{O}$.

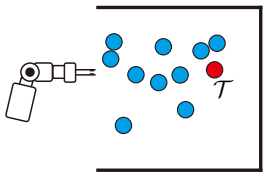


Fig. 2. Illustration of the workspace $\mathcal{W}$ and the robot arm for the same configuration as in Fig. 1, where $\mathcal{T}$ is the target object.

The configuration $\mathcal{X}$ is defined as $\mathcal{X} = \{\mathcal{P}, p_\mathcal{T}, s\} = \{(p_1, \ldots, p_n), p_\mathcal{T}, s\} \in \mathcal{W}^{n+1} \times SE(2)$, where $p_i \in \mathcal{W}$ defines the position of $o_i$, i.e., the coordinates of $o_i$'s centroid, $p_\mathcal{T}$ defines the position of $\mathcal{T}$, and $s$ is the position and orientation of the robot's gripper $g$. $\mathcal{X}[o_i] = p_i$ indicates that the object $o_i$ assumes the position $p_i$, $\mathcal{X}[\mathcal{T}] = p_\mathcal{T}$ means that $\mathcal{T}$ is at position $p_\mathcal{T}$ while $\mathcal{X}[g] = s$ indicates that the gripper $g$ of the robot is at pose $s \in SE(2)$. Define as $V(p)$ the subset of $\mathcal{W}$ occupied by an object at position $p$.

A configuration $\mathcal{X}$ is *feasible* if no object-object collisions occurs, i.e., $\mathcal{X}$ is feasible if $\forall i, j \in [1, n], i \neq j : V(\mathcal{X}[o_i]) \cap V(\mathcal{X}[o_j]) = \emptyset$ and $\forall i \in [1, n] : V(\mathcal{X}[o_i]) \cap V(\mathcal{X}[\mathcal{T}]) = \emptyset$. Given a feasible configuration $\mathcal{X}$, the arm can push multiple

objects at a time. For an object $o_i$ in the set $I \subset \mathcal{O}$ of objects affected by the push, the object is relocated from its current position $p_i = \mathcal{X}[o_i]$ to a new position $p'_i$, giving rise to a new configuration $\mathcal{X}'$, where $\mathcal{X}'[o_i] = p'_i$ and $\forall j \in [1, n], o_j \notin I : \mathcal{X}'[o_j] = \mathcal{X}[o_j]$. The arm's motion results in continuous paths $\pi : [0, 1] \to \mathcal{W}$ with $\pi(0) = c$ and $\pi(1) = c'$, where $c$ and $c'$ are positions in $\mathcal{W}$ obtained by the methods described below.

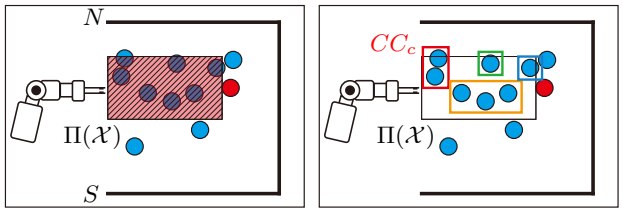


Fig. 3. (left) The path region $\Pi(\mathcal{X})$. (right) Connected components in $\Pi(\mathcal{X})$ using $r = 0.086$, where $CC_c = CC_c(\mathcal{X}, r)$ is the closest connected component to the gripper.

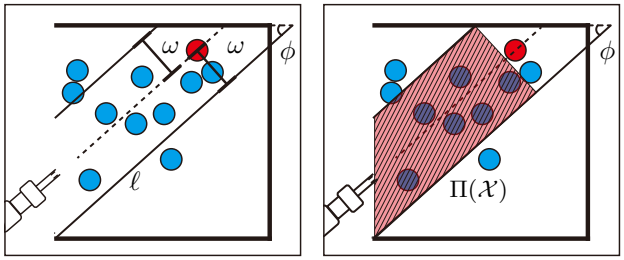


Fig. 4. Path region $\Pi(\mathcal{X})$ with an incidence angle $\phi$, where $w$ is the width of the arm and $\ell$ is the straight line, which passes through $(0, 0)$ and below $\mathcal{X}[\mathcal{T}] = p_{\mathcal{T}}$ with $d(p, \ell) = w$.

## IV. Method

Persistent homology is a tool from algebraic topology that computes topological features of a space at different spatial resolutions. It has been extensively applied for topological analysis of point-cloud data (e.g., see [31] and [32]). Given a collection of points, consider growing balls of radius $r$ centered at each point. As the balls expand, track the unions of all these balls as they overlap each other given a 1-parameter family of spaces. For each radius $r$, one can build a Vietoris–Rips complex (abstract simplicial complex) using the information given by the intersection of the $r$-balls (for more details see [32]). In this setting, persistent homology is the homology of the Vietoris–Rips complex as a function of $r$. Intuitively, persistent homology counts the number of connected components (clusters in our setup) and holes of various dimensions and keeps track of how they change with parameters.

Since we are interested in computing connected components that persist, we focus on the zeroth-homology (zero dimensional homology) and use the field $\mathbb{Z}_2$ as a coefficient for the homology. As the radius $r$ increases, the zero dimensional persistent homology records when the ball in one connected component first intersects a ball of a different connected component, merging both connected components in one, see Figs. 5 and 6.

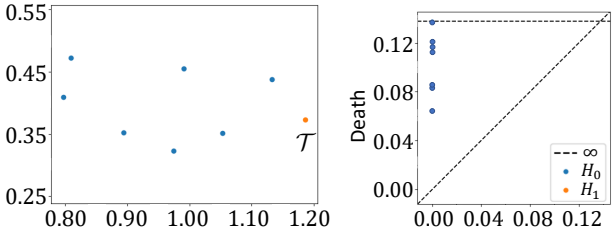


Fig. 5. (left) The subset of objects in the path region $\Pi(\mathcal{X})$ for the problem in Fig. 1. Blue ones are obstacles and the orange one (rightmost point) is the target $\mathcal{T}$. (right) The persistence diagram for the points shown in the left figure.

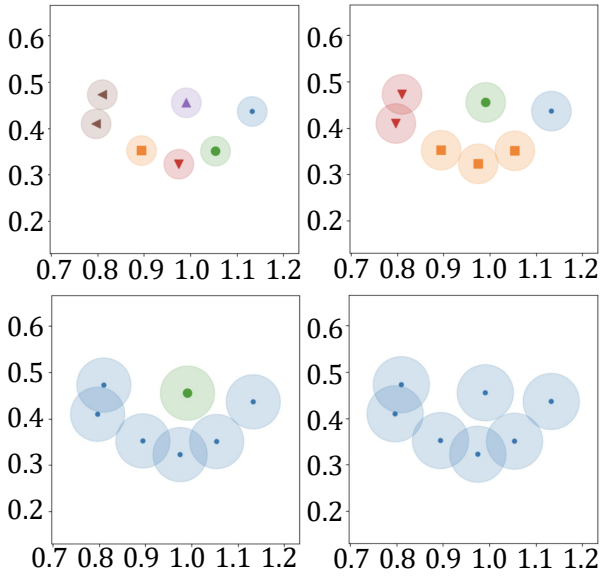


Fig. 6. Examples of connected components that persist obtained for four different radii $r$ in the persistence diagram in Fig. 5. (upper-left) For $r = 0.064$, there are 6 different connected components shown by different markers. (upper-right) For $r = 0.086$, there are 4 connected components. (bottom-left) For $r = 0.121$, two connected components. (bottom-right) Only one connected component (it contains all points) for $r \geq 0.14$.

### A. Proposed Approach for Non-Prehensile Manipulation

Given a target $\mathcal{T} \in \mathcal{O}$, we can define a region $\Pi(\mathcal{X}) \subset \mathcal{W}$ smaller than the workspace where we use persistent homology to find connected components and perform pushing actions. We denote $\Pi(\mathcal{X})$ as the *path region*, where the robot arm will push the obstacles to clean the path to the target. The shape and size of $\Pi(\mathcal{X})$ depends on the boundary of $\mathcal{W}$ and the width $w$ of the robot arm. Since $\mathcal{W}$ is rectangular shelf, hence it is enough to use $N \subset \mathcal{W}$ and $S \subset \mathcal{W}$ (the parallel walls of the shelf) to describe $\Pi(\mathcal{X})$ by using Algorithm 1 and the ideas shown in Figs. 3 and 4.

Fix an orientation of the shelf where the axis $x$ corresponds to the depth of the shelf and the axis $y$ is aligned with the width of shelf with $S$ in $y = 0$ and $N$ in $y = w_s$, where $w_s$ is the width of the shelf. When the Euclidean distance between $\mathcal{X}[\mathcal{T}] = p_{\mathcal{T}}$ and $S \cup N$ is greater than $w$, $d(p_{\mathcal{T}}, S \cup N) > w$, $\Pi(\mathcal{X})$ is a rectangle given by the points $(\mathcal{X}[g]_x, \mathcal{X}[\mathcal{T}]_y - w)$ and $(\mathcal{X}[\mathcal{T}]_x, \mathcal{X}[\mathcal{T}]_y + w)$, where $\mathcal{X}[\mathcal{T}] = (\mathcal{X}[\mathcal{T}]_x, \mathcal{X}[\mathcal{T}]_y)$ and $\mathcal{X}[g] = (\mathcal{X}[g]_x, \mathcal{X}[g]_y)$. See Fig. 3 for an illustration.

For the case where $d(p_{\mathcal{T}}, S \cup N) \leq w$, the path region $\Pi(\mathcal{X})$ has to have an incidence angle $\phi$ with respect to $S$ or $N$, the closest to $p_{\mathcal{T}}$. Without loss of generality, assume that $d(p_{\mathcal{T}}, N) < w$, then $\phi$ is the acute angle between $y = w_s$ and straight line $\ell$ that passed through $(0, 0)$ with $d(p_{\mathcal{T}}, \ell) = w$, such that $\mathcal{X}[\mathcal{T}]$ is above $\ell$. See Figs. 4 and 8. We use the planar rotation matrix $R_{-\phi}$ to rotate $\mathcal{W}$ to obtain a rotated path region $R_{-\phi}(\Pi(\mathcal{X}))$, such that it is analogous to the rectangle found for the case $d(p_{\mathcal{T}}, S \cup N) > w$.

**Algorithm 1:** Path Region($\mathcal{X}$)

$\Pi(\mathcal{X}) \leftarrow \emptyset$
**if** $d(p, S \cup N) > w$ **then**
  **for** $o \in \mathcal{O}_b$ **do**
    **if** $\mathcal{X}[\mathcal{T}]_y - w \leq \mathcal{X}[o]_y \leq \mathcal{X}[\mathcal{T}]_y + w$ *and* $\mathcal{X}[g]_x \leq \mathcal{X}[o]_x < \mathcal{X}[\mathcal{T}]_x$ **then**
      $\Pi(\mathcal{X}) \leftarrow o$
**else**
  **for** $o \in \mathcal{O}_b$ **do**
    **if** $(R_{-\phi}\mathcal{X}[\mathcal{T}]^T)_y - w \leq (R_{-\phi}\mathcal{X}[o]^T)_y$, $(R_{-\phi}\mathcal{X}[o]^T)_y \leq (R_{-\phi}\mathcal{X}[\mathcal{T}]^T)_y + w$, $(R_{-\phi}\mathcal{X}[g]^T)_x \leq (R_{-\phi}\mathcal{X}[o]^T)_x$, *and* $(R_{-\phi}\mathcal{X}[o]^T)_x < (R_{-\phi}\mathcal{X}[\mathcal{T}]^T)_x$ **then**
      $\Pi(\mathcal{X}) \leftarrow o$
**return** $\Pi(\mathcal{X})$

Let $\mathcal{CC}(\mathcal{X}, r)$ be the collection of connected components for the radius $r > 0$ inside of $\Pi(\mathcal{X})$. We use the software Ripser [33] to find the zero dimensional persistent homology, and then to extract the connected component and its generators. For our setting, we only need the closest connected component, $\mathcal{CC}(\mathcal{X}, r)_c$, to the position of the gripper of the robot arm $\mathcal{X}[g]$, since the arm will try to push all obstacles in $\mathcal{CC}(\mathcal{X}, r)_c$ to the outside of $\Pi(\mathcal{X})$. Such manipulation of objects is non-prehensile, so it is not guaranteed that the new arrangement $\mathcal{X}$ of $\mathcal{O}$ will preserve all connected components in $\Pi(\mathcal{X})$.

Now, we compute the circumscribed rectangle, $\mathcal{R}(\mathcal{X}, r)$, of $\mathcal{CC}(\mathcal{X}, r)_c$ (the smallest rectangle that contains $\mathcal{CC}(\mathcal{X}, r)_c$, such that the sides of $\mathcal{R}(\mathcal{X}, r)$ are parallel to the $x$ and $y$ axes). We use it to plan the pushing actions to move all obstacles $\mathcal{CC}(\mathcal{X}, r)_c$ with one single action. Algorithm 2 CRCCC (Circumscribed Rectangle of the Closest Connected Components) describes this procedure.

**Algorithm 2:** CRCCC($\mathcal{X}, r$)

$\Pi(\mathcal{X}) \leftarrow$ `PathRegion`($\mathcal{X}$)
$\mathcal{CC}(\mathcal{X}, r) \leftarrow$ `Ripser`($\Pi(\mathcal{X})$, $r$, dim $= 0$)
$\mathcal{CC}(\mathcal{X}, r)_c \leftarrow$ `ClosestSet`($\mathcal{CC}(\mathcal{X}, r)$, $s$)
$\mathcal{R}(\mathcal{X}, r) \leftarrow$ `CircumscribedRectangle`($\mathcal{CC}(\mathcal{X}, r)_c$)
**return** $\mathcal{R}(\mathcal{X}, r)$

With the circumscribed rectangle of the closest connected components $\mathcal{R}(\mathcal{X}, r)$ the arm will push the region described by $\mathcal{R}(\mathcal{X}, r)$ either to up or down. It depends on the location of $\mathcal{R}(\mathcal{X}, r)$ relative to the path region $\Pi(\mathcal{X})$, if the $y$ coordinate of the centroid of $\mathcal{R}(\mathcal{X}, r)$, $c_{\mathcal{R}(\mathcal{X},r)y}$, is higher than the $y$ coordinate of the centroid of $\Pi(\mathcal{X})$, $c_{\Pi(\mathcal{X})y}$. This will decide whether the arm will do a sweeping movement from the bottom of $\mathcal{R}(\mathcal{X}, r)$ to top until the arm reaches

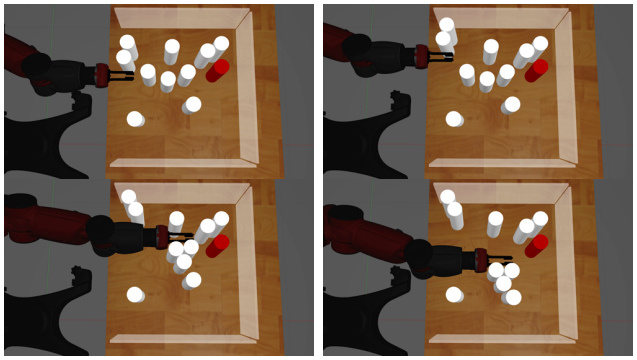

Fig. 7. Sequence of pushing actions, where the selected radius $r_m = 0.086$ for the first action is the minimal of $R = R_{\nu,h}$ for $\nu = 0.015$ and $h = 0.08$. Such radius $r_m$ produces the connected components described in Fig. 6 with $r = 0.086$.

$N$, the north boundary of $\Pi(\mathcal{X})$. The opposite is executed if $c_{\mathcal{R}(\mathcal{X},r)y} < c_{\Pi(\mathcal{X})y}$ . For the case where $c_{\mathcal{R}(\mathcal{X},r)y} = c_{\Pi(\mathcal{X})y}$ perform the same analysis with $c_{\mathcal{R}(\mathcal{X},r)y}$ and the $y$ coordinate of the centroid of $\mathcal{W}$, $c_{\mathcal{W}y}$. See Algorithm 3 for details.

**Algorithm 3:** Pushing Actions($\mathcal{X}, r$)

$\mathcal{R}(\mathcal{X}, r) \leftarrow$ `CRCCC`($\mathcal{X}, r$)
**if** $c_{\mathcal{R}(\mathcal{X},r)y} > c_{\Pi(\mathcal{X})y}$ **then**
  `SweepingBottomToTop()`
**else**
  **if** $c_{\mathcal{R}(\mathcal{X},r)y} < c_{\Pi(\mathcal{X})y}$ **then**
    `SweepingTopToBottom()`
  **else**
    **if** $c_{\mathcal{R}(\mathcal{X},r)y} \leq c_{\mathcal{W}y}$ **then**
      `SweepingTopToBottom()`
    **else**
      `SweepingBottomToTop()`
$\mathcal{X} \leftarrow$ `UpdateConfiguration()`
$t \leftarrow$ `ActionsTime()`
**return** *($\mathcal{X}$, t)*

Up this point we have computed everything for a given radius $r$. We use the persistence diagram to select the appropriate $r$ in an informed manner. First, define the *persistent radius* for a given value $\nu > 0$ as a radius $r$ where one or more connected components die. Between $r$ and $r + \nu$ the number of connected components remains the same, in other words the connected components persist. Let $R_\nu$ be the set of all persistent radii for a given value $\nu > 0$. In the persistence diagram in Fig. 5, $R_{0.015} = \{0.064, 0.086, 0.121, 0.14\}$. Note that, $R_h$ is not empty since the radius where the last connected component dies belongs to $R_\nu$.

For our setup, $\nu$ is considered as the margin of error for the pushing action when the arm is moving a whole connected component outside the path region $\Pi(\mathcal{X})$.

Observe that $R_\nu$ may contain small radii such that the gripper cannot move between two connected components, hence we only consider persistent radii greater than $h$, the number given by 110% of width of the gripper (10% more to consider imprecision) plus two times the radius of the objects. Define $R(\mathcal{X}) = R_{\nu,h}(\mathcal{X})$ as the collection of persistent radii greater than $h$ for a configuration space $\mathcal{X}$. For example, in the persistence diagram of Fig. 5, $R_{\nu,h}(\mathcal{X}) = \{0.086, 0.121, 0.14\}$ for $\nu = 0.015$ and $h = 0.08$.

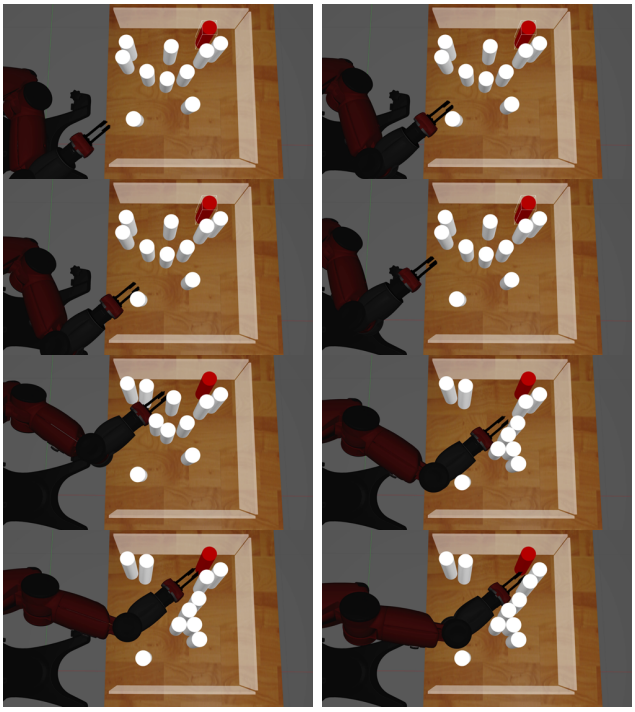

Fig. 8. Sequence of pushing actions with angle $\phi = 0.709$, where the selected radius for the first action is $r_m = 0.091$ with $\nu = 0.015$ and $h = 0.08$.

The simplest approach to solve the reaching through clutter problem is to select $r_m$ as the minimum element of $R(\mathcal{X})$ for each configuration space after each pushing action. See Algorithm 4 for the steps.

**Algorithm 4:** PH Informed Actions($\mathcal{X}_0, \nu, h$)

1 $\mathcal{X} \leftarrow \mathcal{X}_0$
2 **while** $\Pi(\mathcal{X})$ *is not empty* **do**
3 $D_0 \leftarrow$ `Ripser`$(\Pi(\mathcal{X}), dim = 0)$
4 \\ 0d persistence diagram
5 $R(\mathcal{X}) \leftarrow$ `SetOfPersistentRadii`$(D_0, \nu, h)$
6 $r_m \leftarrow$ `MinimumElement`$(R(\mathcal{X}))$
7 $\mathcal{X}' \leftarrow$ `PushingActions`$(\mathcal{X}, r_m)$
8 **if** $\mathcal{X}' = \mathcal{X}$ **then**
9 label $\mathcal{X}$ failure

Another approach to the problem is to use not only $r_m$ but all radii in $R_{\nu,h}(\mathcal{X})$ for a configuration space $\mathcal{X}$. Specifically, it is possible to build a tree with nodes corresponding to configurations in $\mathcal{X}$ and propagate it by performing the push action for each radius in $R_{\nu,h}(\mathcal{X})$. The propagation is performed until it achieves success or failure. Finally, it searches for the path that leads to success and minimizes time for performing the pushing actions as in Algorithm 5.

## V. Experiments

Experiments were run on a Ubuntu workstation with *Intel Core i5-8259U 3.8Ghz 16GB RAM*. Gazebo [34] was used for simulating the task execution and MoveIt [35] was used for forming motion planning queries to the Bi-EST planner [36] in OMPL [37].

For the comparison experiments of the GRTC-Heuristic, a simplified version of the algorithm in [3] was used that

**Algorithm 5:** PH Informed Search($\mathcal{X}_0, \nu, h$)

1 tree $\leftarrow$ {nodes $= \{\mathcal{X}_0\}, \emptyset\}$
2 $Q \leftarrow \{\mathcal{X}_0\}$
3 **while** $Q \neq \emptyset$ **do**
4 $Q' \leftarrow \emptyset$
5 **for** $\mathcal{X} \in Q$ **do**
6 **if** $\Pi(\mathcal{X}) = \emptyset$ **then**
7 label $\mathcal{X}$ success **continue**
8 $D_0 \leftarrow$ `Ripser`$(\Pi(\mathcal{X}), dim = 0)$
9 $R(\mathcal{X}) \leftarrow$ `SetOfPersistentRadii`$(D_0, \nu, h)$
10 **for** $r \in R(\mathcal{X})$ **do**
11 $(\mathcal{X}_{new}, t_{new}) \leftarrow$ `PushActions`$(\mathcal{X}, r)$
12 **if** $\mathcal{X} = \mathcal{X}_{new}$ **then**
13 label $\mathcal{X}$ fail **continue**
14 tree.nodes = tree.nodes$\cup\{\mathcal{X}_{new}\}$
15 tree.edges = tree.edges$\cup\{(\mathcal{X}, \mathcal{X}_{new}), t\}$
16 $Q' = Q' \cup \{\mathcal{X}_{new}\}$
17 $Q \leftarrow Q'$
18 path $\leftarrow$ `ShortestSuccessfulPath`(tree)

pushes cylinders in a straight line to their goal region. The straight pushes are generated using a kinematic motion planner. This varies from the original GRTC-Heuristic, which uses a kinodynamic motion planner to randomly sample push actions to get the cylinder into a goal region.

We also run experiments using a version of the algorithm PH Informed Actions 4 where we remove the persistent homology information. More specifically, the lines 3-6 are changed by $r_m \leftarrow 0.01$, that is, an algorithm that does not use persistent homology and does not group obstacles. We denote this algorithm by OOA (one by one action). The idea is to compare the statistical difference between non-prehensile manipulation without grouping the obstacles and the PH informed actions.

For all methods, an overall planning time threshold of 300 seconds was set. If a method exceeds this threshold, it exits and returns a failure to retrieve the assigned object. To evaluate performance of the proposed algorithms we conduct 121 experiments: one manually designed (in Fig. 1); 10 instances from a prior work [3]; 10 randomly generated instances that are simple (with only 4 objects such that all are far from the wall); and 100 random instances where the target object is always behind the obstacles and deep in the shelf. Some scenes are presented in Fig. 9.

Fig. 10 summarizes the results of the experiments for the scenes with three obstacles (easy case, see Fig. 9). We have decided to compare all algorithms with a less cluttered environment in order to increase the success rate of the GRTC-Heuristic. The overall success rate is high as expected. The planning time for GRTC-Heuristic, however, is still high since it samples points to perform the pushing action.

The comparison results with the scenes provided in related work [3] is shown in Fig. 11. The success rate for GRTC-Heuristic is low and it agrees with the experiments shown in the related work [3]. Note that the number of actions for the algorithm OOA may be high but it eventually achieves success. When the number of actions is equal between PHIA and OOA, it is possible to see that the planning time for PHIA is slightly higher than OOA. The reason behind this

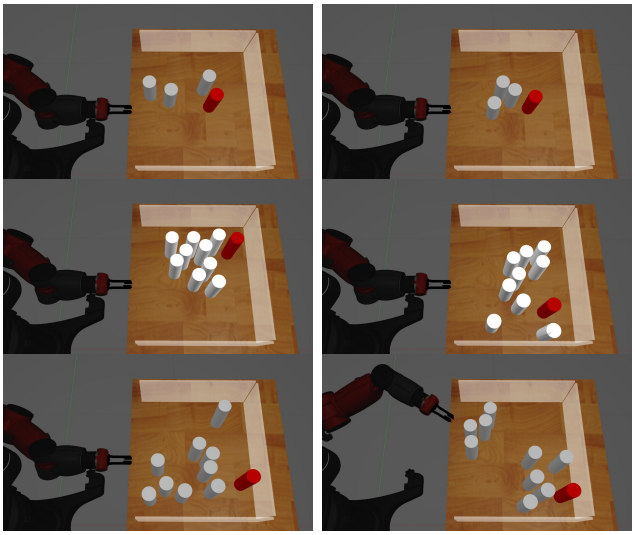

Fig. 9. From left to right: (first) simple case number 1; (second) simple case number 7; (third) S1 case from [3]; (forth) S7 case from [3]; (fifth) random experiment number 3; (sixth) random experiment number 32.

difference is the computation time to perform the topological data analysis. Nevertheless, when the number of actions is different between PHIA and OOA, the planning time for OOA increases since for each configuration $\mathcal{X}$ it has to find the closest obstacle to be pushed away from the path region $\Pi(\mathcal{X})$.

The results from the 100 random scenes shown in Fig. 12 further confirm the conclusions presented in previous paragraphs. The average of actions proposed by GRTC-heuristic is shown only for the successful cases. Observe that the average number of actions performed by PHIS is lower than the other methods. This comes, however, with a significant increase in the planning time but not enough to pass the planning time of the GRTC-heuristic. In summary, the conclusion we can draw from the experiments is that PHIA has the best planning time and PHIS is the best in terms of number of actions.

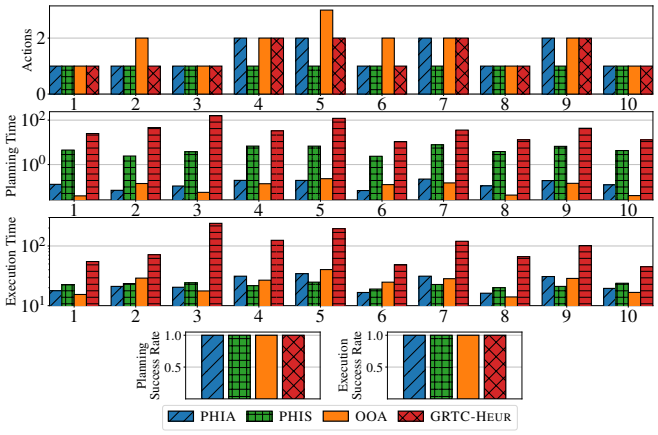


Fig. 10. Ten simulations examples for the simple case where the number of objects is four, see top row of Fig. 9.

## VI. Conclusion and Future Work

In this study, we presented an application of topological data analysis for non-prehensile manipulations in clutter. Our

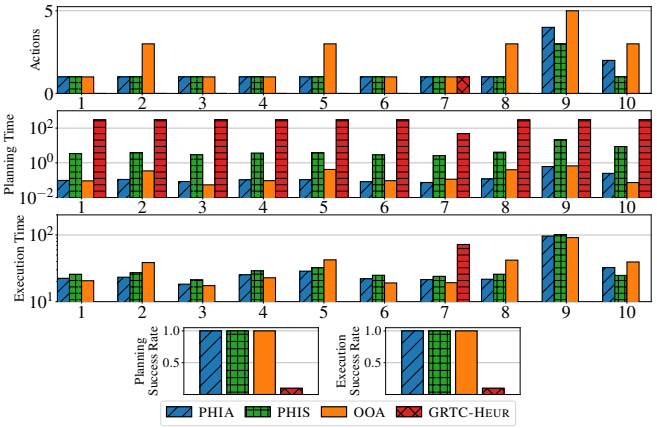


Fig. 11. Ten simulation examples from a prior work [3]. Execution time is always zero when the planning time reaches 300s.

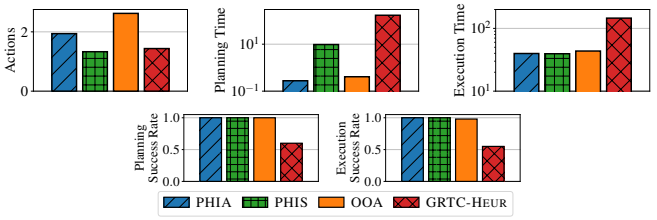


Fig. 12. One hundred random cases, where each bar represents the average of the 100 random instances for each algorithm.

simulated experiments show that the algorithms based on persistent homology have higher success and are faster in finding solutions than the baselines and alternatives from the literature. The experiments indicate that the topological data analysis is cheaper to compute. Its use did not notably increase planning time. Instead, it decreased planning time by reducing the number of actions needed.

The success of the experiments motivates further exploration of the topological method for non-prehensile manipulation. For instance, we simplified the problem by considering objects of cylindrical shape for experiments; however, the framework can be applied to more general objects given further analysis. It is straightforward to see that the persistent homology approach developed in this paper can be readily applied to objects of different shapes and sizes; it will be interesting to explore how our algorithm adapt to different object shape distributions. Because our approach has very fast planning time, it would also be very interesting to develop it to enable continuous decision making, instead of taking step-by-step actions.

Follow up work focuses on how to realize the proposed framework on a real Baxter robot. This calls for consideration of uncertainty that may arise from many sources including perception and robot motion. Since the persistent connected components are robust and can be described by the parameter $\nu$, the approach is promising for planning pushing actions when the uncertainty of the pose of the objects is high but bounded. Different types of manipulations may be useful to consider as well, such as pick and place actions to separate highly dense clusters of objects.

## REFERENCES


[1] M. R. Dogar, K. Hsiao, M. Ciocarlie, and S. S. Srinivasa, "Physics-based grasp planning through clutter," *Robotics: Science and Systems VIII*, p. 57, 2013.

[2] W. Bejjani, R. Papallas, M. Leonetti, and M. R. Dogar, "Planning with a receding horizon for manipulation in clutter using a learned value function," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2018, pp. 1–9.

[3] R. Papallas and M. R. Dogar, "Non-prehensile manipulation in clutter with human-in-the-loop," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 6723–6729.

[4] S. D. Han, N. M. Stiffler, A. Krontiris, K. E. Bekris, and J. Yu, "Complexity results and fast methods for optimal tabletop rearrangement with overhand grasps," *The International Journal of Robotics Research*, vol. 37, no. 13-14, pp. 1775–1795, 2018.

[5] R. Wang, K. Gao, D. Nakhimovich, J. Yu, and K. E. Bekris, "Uniform object rearrangement: From complete monotone primitives to efficient non-monotone informed search," *arXiv preprint arXiv:2101.12241*, 2021.

[6] Y. Labbé, S. Zagoruyko, I. Kalevatykh, I. Laptev, J. Carpentier, M. Aubry, and J. Sivic, "Monte-carlo tree search for efficient visually guided rearrangement planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3715–3722, 2020.

[7] S. D. Han, S. W. Feng, and J. Yu, "Toward fast and optimal robotic pick-and-place on a moving conveyor," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 446–453, 2019.

[8] E. Huang, Z. Jia, and M. T. Mason, "Large-scale multi-object rearrangement," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 211–218.

[9] H. Song, J. A. Haustein, W. Yuan, K. Hang, M. Y. Wang, D. Kragic, and J. A. Stork, "Multi-object rearrangement with monte carlo tree search: A case study on planar nonprehensile sorting," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 9433–9440.

[10] M. Nieuwenhuisen, D. Droeschel, D. Holz, J. Stückler, A. Berner, J. Li, R. Klein, and S. Behnke, "Mobile bin picking with an anthropomorphic service robot," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 2327–2334.

[11] R. Shome, W. N. Tang, C. Song, C. Mitash, H. Kourtev, J. Yu, A. Boularias, and K. E. Bekris, "Towards robust product packing with a minimalistic end-effector," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9007–9013.

[12] C. Song and A. Boularias, "Object rearrangement with nested non-prehensile manipulation actions," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 6578–6585.

[13] Z. Pan and K. Hauser, "Decision making in joint push-grasp action space for large-scale object sorting," *arXiv preprint arXiv:2010.10064*, 2020.

[14] G. Havur, G. Ozbilgin, E. Erdem, and V. Patoglu, "Geometric rearrangement of multiple movable objects on cluttered surfaces: A hybrid reasoning approach," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 445–452.

[15] J. Zhou, Y. Hou, and M. T. Mason, "Pushing revisited: Differential flatness, trajectory planning, and stabilization," *The International Journal of Robotics Research*, vol. 38, no. 12-13, pp. 1477–1489, 2019.

[16] J. Zhou, M. T. Mason, R. Paolini, and D. Bagnell, "A convex polynomial model for planar sliding mechanics: theory, application, and experimental validation," *The International Journal of Robotics Research*, vol. 37, no. 2-3, pp. 249–265, 2018.

[17] B. Huang, S. D. Han, A. Boularias, and J. Yu, "Dipn: Deep interaction prediction network with application to clutter removal," in *IEEE International Conference on Robotics and Automation*, 2021.

[18] J. Lee, Y. Cho, C. Nam, J. Park, and C. Kim, "Efficient obstacle rearrangement for object manipulation tasks in cluttered environments," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 183–189.

[19] C. Nam, J. Lee, Y. Cho, J. Lee, D. H. Kim, and C. Kim, "Planning for target retrieval using a robotic manipulator in cluttered and occluded environments," *arXiv preprint arXiv:1907.03956*, 2019.

[20] J. A. Haustein, J. King, S. S. Srinivasa, and T. Asfour, "Kinodynamic randomized rearrangement planning via dynamic transitions between statically stable states," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 3075–3082.

[21] A. Cosgun, T. Hermans, V. Emeli, and M. Stilman, "Push planning for object placement on cluttered table surfaces," in *2011 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2011, pp. 4627–4632.

[22] W. Yuan, J. A. Stork, D. Kragic, M. Y. Wang, and K. Hang, "Rearrangement with nonprehensile manipulation using deep reinforcement learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 270–277.

[23] J. A. Haustein, I. Arnekvist, J. Stork, K. Hang, and D. Kragic, "Learning manipulation states and actions for efficient non-prehensile rearrangement planning," *arXiv preprint arXiv:1901.03557*, 2019.

[24] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, "Manipulation planning on constraint manifolds," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 625–632.

[25] A. H. Qureshi, J. Dong, A. Choe, and M. C. Yip, "Neural manipulation planning on constraint manifolds," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6089–6096, 2020.

[26] P. Englert, I. M. R. Fernández, R. K. Ramachandran, and G. S. Sukhatme, "Sampling-based motion planning on sequenced manifolds," *arXiv preprint arXiv:2006.02027*, 2020.

[27] J. Basch, L. J. Guibas, D. Hsu, and A. T. Nguyen, "Disconnection proofs for motion planning," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, vol. 2. IEEE, 2001, pp. 1765–1772.

[28] Z. McCarthy, T. Bretl, and S. Hutchinson, "Proving path non-existence using sampling and alpha shapes," in *2012 IEEE international conference on robotics and automation*. IEEE, 2012, pp. 2563–2569.

[29] S. Bhattacharya, R. Ghrist, and V. Kumar, "Persistent homology for path planning in uncertain environments," *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 578–590, 2015.

[30] F. T. Pokorny, M. Hawasly, and S. Ramamoorthy, "Topological trajectory classification with filtrations of simplicial complexes and persistent homology," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 204–223, 2016.

[31] H. Edelsbrunner, *A short course in computational geometry and topology*, ser. SpringerBriefs in Applied Sciences and Technology. Springer, Cham, 2014. [Online]. Available: https://doi.org/10.1007/978-3-319-05957-0

[32] T. Kaczynski, K. M. Mischaikow, M. Mrozek, and K. Mischaikow, *Computational homology / Tomasz Kaczynski, Konstantin Mischaikow, Marian Mrozek.*, ser. Applied mathematical sciences (Springer-Verlag New York Inc.); v. 157. New York: Springer, 2004.

[33] C. Tralie, N. Saul, and R. Bar-On, "Ripser.py: A lean persistent homology library for python," *The Journal of Open Source Software*, vol. 3, no. 29, p. 925, Sep 2018. [Online]. Available: https://doi.org/10.21105/joss.00925

[34] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, pp. 2149–2154 vol.3.

[35] D. Coleman, I. A. Sucan, S. Chitta, and N. Correll, "Reducing the barrier to entry of complex robotic software: a moveit! case study," *CoRR*, vol. abs/1404.3785, 2014. [Online]. Available: http://arxiv.org/abs/1404.3785

[36] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *Proceedings of International Conference on Robotics and Automation*, vol. 3. IEEE, 1997, pp. 2719–2726.

[37] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, https://ompl.kavrakilab.org.