Data-Driven Topological Motion Planning with Persistent Cohomology

Florian T. Pokorny and Danica Kragic
Centre for Autonomous Systems, KTH Royal Institute of Technology

***Abstract*— In this work, we present an approach to topological motion planning which is fully data-driven in nature and which relies solely on the knowledge of samples in the free configuration space. For this purpose, we discuss the use of persistent cohomology with coefficients in a finite field to compute a basis which allows us to efficiently solve the path planning problem. The proposed approach can be used both in the case where a part of a configuration space is well-approximated by samples and, more generally, with arbitrary filtrations arising from real-world data sets. Furthermore, our approach can generate motions in a subset of the configuration space specified by the sub- or superlevel set of a filtration function such as a cost function or probability distribution. Our experiments show that our approach is highly scalable in low dimensions and we present results on simulated PR2 arm motions as well as GPS trace and motion capture data.**

I. Introduction and Related Work

In recent years, the "data-driven approach" has provided a paradigm shift in robotics: instead of hand-coded mathematical models of an idealized version of the environment, a multitude of techniques which build probabilistic models from sampled data have been developed [1]. Within the motion planning community, this development has been mirrored in the development of sampling based motion planning techniques such as rapidly exploring random trees (RRTs) [2] and probabilistic roadmaps (PRMs) [3]. In this work we consider how to determine not just a single collision free trajectory between two points, but **a collection of homotopy inequivalent trajectories given collision free samples**, where trajectories are called homotopy inequivalent when they cannot be continuously deformed into one another without collisions. While motion planning approaches such as RRT and PRM-based methods [2], [3] aim to generate either just a collision-free path, or a path which is close to optimal with respect to some cost, the underlying graphs used in these approaches are only able to approximate the path-connectivity of the free configuration space $\mathcal{C}_f$ and cannot capture higher order topological features such as the number of holes and voids. In particular, they do not capture information about the homology or homotopy groups of $\mathcal{C}_f$ besides path connectivity.

On the other hand, while classical analytical methods to motion planning, such as the works of [4] and [5], [6] construct cell complexes which can be used to compute homology groups of $\mathcal{C}_f$ to extract this type of information, these methods have not focused on generating trajectories in distinct homotopy classes and assumed that a complete and noise-free description of $\mathcal{C}_f$ is specified, for example in terms of semi-algebraic functions.

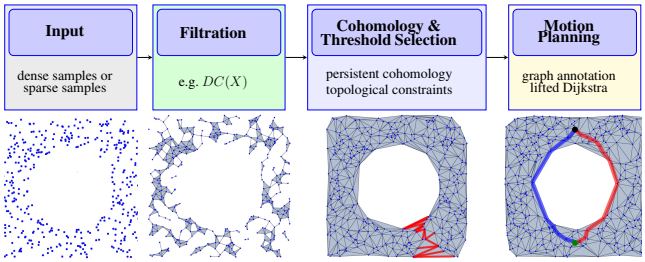

Fig. 1: Overview of our data-driven topological motion planning approach.

A small number of recent works, including [7], [8], [9], [10], [11], have started to study motion planning algorithms for trajectories with homotopy class constraints. In the case of 2D configuration spaces, the cited works rely on differential forms computing the winding angle around representative points inside the obstacles in the configuration space $\mathcal{C}$ and this idea has been generalized to higher dimensions. However, all these works rely on a specific basis of the first de Rham cohomology group in order to perform motion planning and they require an explicit geometric reduction of the obstacles in $\mathcal{C}_f$ to representative skeleta. Our approach, as outlined in Fig. 1, instead assumes only the availability of samples in $\mathcal{C}_f$, obtained using random sampling, or from a database of successfully executed prior motions. From these samples, a simplicial complex filtration is then extracted, taking into account a distance, probability density, or cost function. This part of our approach is closely related to [12] which introduced persistent homology as a tool for *classifying pre-existing trajectories*, but which did not consider the *motion planning problem* itself. We utilize persistent cohomology rather than homology to determine 1-cocycles representing large features in the filtration and to find a filtration threshold for motion planning. The resulting 1-cocycles correspond to 'fences' in the configuration space using which we are able to formulate the topological motion planning problem by means of a cohomologically annotated graph serving as input to an efficient implicit graph representation and a lifted Dijkstra algorithm we introduce here.

II. Background and Notation

Algebraic Topology Preliminaries: In this work, we make use of cohomology with coefficients in a field $\mathbb{F}$. While infinite fields such as $\mathbb{Q}$ can be used for all persistent cohomology computations, the path planning algorithm we propose performs a search over a finite search space only in the case when $\mathbb{F}$ is also finite. Most typically, we shall consider the finite field $\mathbb{Z}_p$ with p elements, where p is a prime and $\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$. The binary field $\mathbb{Z}_2$ will

Symbol	Explanation	Symbol	Explanation
$\mathcal{C}$	configuration space	$\mathcal{C}_f$	collision free c-space
$\mathcal{K}$	simplicial complex	$\|\mathcal{K}\|$	support of $\mathcal{K}$
$C_p(\mathcal{K})$	p-chains of $\mathcal{K}$	$C^p(\mathcal{K})$	p-cochains of $\mathcal{K}$
∂_p	pth boundary operator	δ_p	pth coboundary operator
$B_p(\mathcal{K})$	p-boundaries of $\mathcal{K}$	$B^p(\mathcal{K})$	p-coboundaries of $\mathcal{K}$
$Z_p(\mathcal{K})$	p-cycles of $\mathcal{K}$	$Z^p(\mathcal{K})$	p-cocycles of $\mathcal{K}$
$H_p(\mathcal{K})$	pth homology group	$H^p(\mathcal{K})$	pth cohomology group
$H_p^{i,j}(\mathcal{K})$	pth (i,j) persistent homology group	$H^p_{i,j}(\mathcal{K})$	pth (i,j) persistent cohomology group
$\mathbb{F}$	a finite field	$\mathbb{Z}_p$	finite field $\{0,\dots,p-1\}$
$\pi_1(Y)$	1^{st} fundamental group of Y		for some prime p
X	set of samples in $\mathbb{R}^d$	X_r	union of r-balls around X

TABLE I: Summary of our notation

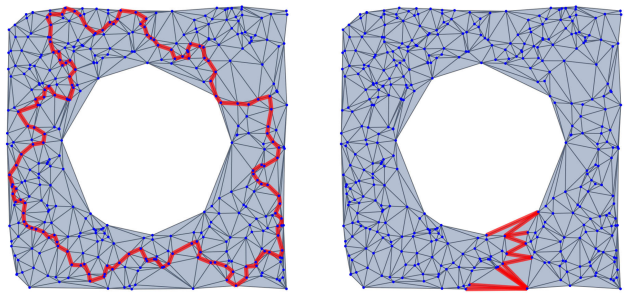

Fig. 2: A simplicial complex $\mathcal{K}$ in 2D, consisting of vertices, edges and triangles is displayed in both parts of the figure. On the left, we display a 1-cycle c in red, such that $[c] \in H_1(\mathcal{K})$ yields a basis for the 1-dimensional $H_1(\mathcal{K})$ with $\mathbb{Z}_2$ coefficients. Marked edges correspond to 1-simplices with coefficient 1 in c. In the right figure, a 1-cochain φ is displayed, where the marked red edges correspond to 1-simplices on which φ takes the value 1, while φ takes the value 0 on the remaining 1-simplices.

be used in our experiments since it can be implemented very efficiently. The books [13], [14] provide a reference for the relevant concepts from algebraic topology, and a summary of our notation is provided in Table I.

Simplicial Complexes: Recall that a geometric k-simplex $\sigma = [v_0, \dots, v_k]$ in $\mathbb{R}^d$ is a convex hull of $k+1$ ordered affinely independent elements $v_0, \dots, v_k \in \mathbb{R}^d$ and a convex hull of an ordered subset of these elements is called a face τ of σ, indicated by $\tau \leqslant \sigma$. We call k the dimension of a k-simplex. In the special case of $\mathbb{Z}_2$ coefficients as in [12], the ordering can in fact be ignored. A (finite) simplicial complex $\mathcal{K}$ is a non-empty set of simplices such that if $\sigma \in \mathcal{K}$ and $\tau \leqslant \sigma$, then $\tau \in \mathcal{K}$ and if $\sigma, \sigma' \in \mathcal{K}$ then $\sigma \cap \sigma'$ is empty or an element of $\mathcal{K}$. We write $|\mathcal{K}|$ for set of points in $\mathbb{R}^d$ contained in the union of all simplices in $\mathcal{K}$. The set $|\mathcal{K}|$ is a topological space with the subspace topology from $\mathbb{R}^d$.

(Co-)Homology with Coefficients in $\mathbb{F}$***:*** A p-chain c is a formal sum $c = \sum_{i=1}^{k} \lambda_i \sigma_i$ of p-simplices $\{\sigma_i\}_{i=1}^k \subset \mathcal{K}$ with $\lambda_i \in \mathbb{F}$ and $C_p(\mathcal{K})$ denotes the $\mathbb{F}$-vector space of all p-chains. In particular, for finite geometric complexes, 1-chains are finite linear combinations of edges and 2-chains are linear combinations of triangles. We denote by $C^p(\mathcal{K}) = Hom(C_p(\mathcal{K}), \mathbb{F})$ the vector space of linear maps from $C_p(\mathcal{K})$ to $\mathbb{F}$. Elements of $C^p(\mathcal{K})$ are called p-cochains. When no confusion arises, we write C_p, C^p for $C_p(\mathcal{K}), C^p(\mathcal{K})$ to simplify notation. For every geometric p-simplex $\sigma = [v_0, \dots, v_p]$ let $\partial\sigma$ be the $p-1$-chain $\partial\sigma = \sum_{i=0}^{p} (-1)^i [v_0, \dots, v_{i-1}, v_{i+1}, \dots, v_p]$. For each $p \in \{0, \dots, d\}$, ∂ extends to a linear map $\partial : C_p \to C_{p-1}$, called the boundary operator. The coboundary operator $\delta : C^{p-1} \to C^p$ is defined dually by $\delta(\omega)(c) = \omega(\partial(c))$, for $\omega \in C^{p-1}$ and $c \in C_p$. A p-chain c such that $c = \partial\omega$ for some $\omega \in C_{p+1}$ is called a p-boundary. And a p-chain c such that $\partial c = 0$ is called a p-cycle. Similarly, a p-cochain $\varphi \in C^p$ such that $\varphi = \delta\eta$ for some $\eta \in C^{p-1}$ is called a p-coboundary and a p-cochain φ such that $\delta\varphi = 0$ is called a p-cocycle. The vector spaces of p-boundaries and p-cycles are denoted B_p and Z_p respectively and the vector spaces of p-cochains and p-cocycles are denoted by B^p and Z^p respectively. The p-th homology group of $\mathcal{K}$ is defined by $H_p(\mathcal{K}) = Z_p/B_p$ and the p-th cohomology group is defined by $H^p(\mathcal{K}) = Z^p/B^p$. For a cycle $c \in Z_p$ (cocycle $\varphi \in Z^p$), we denote by $[c] \in H_p$ ($[\varphi] \in H^p$) the resulting element in homology (cohomology). In the special case of $\mathbb{F} = \mathbb{Z}_2$, we can easily visualize 1-chains as a collection of edges in $\mathcal{K}$ which have non-zero coefficients in the chain. Similarly, a 1-cochain corresponds to a function assigning 0 or 1 to each edge, and we can visualize the 1-cochain by displaying those 1-simplices on which it takes the value 1. See Fig. 2 for an example. Note that each p-cycle c yields an element in H_p, but this representative is only unique up to elements in B_p. Similarly, p-cocycles define elements of H^p only up to B^p.

The importance of homology and cohomology in mathematics arises from the fact that they capture *global topological properties* about the topological space defined by $|\mathcal{K}|$. In particular both homology and cohomology are invariant under continuous deformations of the space $|\mathcal{K}|$ (homotopies of $|\mathcal{K}|$). The universal coefficient theorem [13] in fact asserts that H_p and H^p are dual as vector spaces for field coefficients. In particular, for finite simplicial complexes, $b_p = \dim(H_p) = \dim(H^p)$ is called the p^{th} Betti number and counts the number of connected components (b_0), tunnels (b_1), and higher dimensional voids in $\mathcal{K}$. The left part of Fig. 2 illustrates an example 1-cycle c lying in a simplicial complex $\mathcal{K}$ and forming a basis of $H_1(\mathcal{K})$ which is in this case 1-dimensional and where we pick $\mathbb{F} = \mathbb{Z}_2$ coefficients. $|\mathcal{K}|$ is in fact homotopy equivalent to a circle. We display a 1-cocycle φ in the right part of that figure. Over $\mathbb{Z}_2$, 1-cocycles correspond to 'picket fences' [14]. Here, $\dim(H_1(\mathcal{K})) = \dim(H^1(\mathcal{K})) = 1$ and $[c] \in H_1(\mathcal{K})$ and $[\varphi] \in H^1(\mathcal{K})$ form a basis respectively.

Filtrations: While homology and cohomology are classical concepts [13], we now review the more recent developments of persistent (co-)homology [15], [16], which have lead to very efficient algorithms which can now also be used also for the computation of classical (co-)homology. Persistent (co-)homology is concerned with (co-)homology in a multiscale setting. One of the origins of persistence is the study of the topology of sublevel (or superlevel) sets of a function $f : X \to \mathbb{R}$ defined on a topological space X. Each sublevel set $X_r = f^{-1}((-\infty, r])$ yields a topological space X_r, where $X_r \subseteq X_{r'}$ whenever $r \leqslant r'$. As r increases, homological features can be 'born' and disappear or 'die' as the threshold r increases. Persistence provides a computational mechanism for understanding these changes. To make this precise, we work with a filtration $\mathbb{K}$ of finite simplicial complexes in $\mathbb{R}^d$, by which we mean a sequence

$\mathbb{K} : K_1 \subset K_2 \subset \ldots \subset K_n = K_\infty$ of finite simplicial complexes. Typically, each filtration index i is associated to a real valued filtration value r so that $K_i = f^{-1}(-\infty, r]$. For example, we can assign an arbitrary real value to each vertex of K_n. Then the function $f(\sigma) = \max_{i\in\{0,\ldots,k\}} f(v_i)$ for an arbitrary k-simplex $\sigma = [v_0, \ldots, v_k] \in K_n$ gives rise to a filtration of simplicial complexes when its sublevel sets are considered. When $K_i = f^{-1}((-\infty, r])$ we call r the filtration value associated to the filtration index i.

Delaunay-Čech complexes: To model configuration spaces from a finite sample of collision-free points $X \subset \mathcal{C}_f$, we shall consider the family of union of balls spaces $X_r = \bigcup_{x\in X}\{y \in \mathbb{R}^d : \|x - y\| \leqslant r\}$, for $r \geqslant 0$. For each r, X_r is homotopy equivalent to the Delaunay-Čech complex $DC_r(X)$ [17], which is a simplicial complex defined for any finite set $X \subset \mathbb{R}^d$ where each subset of $d+1$ point is affinely independent. This assumption is generic in that a uniform random sample satisfies this condition with probability one and we can also enforce the condition by an arbitrarily small perturbation of X. Let $D(X)$ denote the simplicial complex corresponding to the Delaunay triangulation of X with simplices defined by $D(X) = \{[v_0, \ldots, v_k] : v_i \in X, \cap_{i=0}^k V_{v_i} \neq \emptyset \text{ for } k \in \{0, 1, \ldots, d\}\}$, where V_x denotes the Voronoi cell containing x. For each k-simplex $\sigma = [v_0, \ldots, v_k] \in D(X)$, define $f(\sigma) = \min\{r : \bigcap_{i=1}^k \mathbb{B}_r(v_i) \neq \emptyset\}$, where $\mathbb{B}_r(x) = \{y \in \mathbb{R}^d : \|x - y\| \leqslant r\}$. The Delaunay-Čech complex $DC_r(X)$, for $r \geqslant 0$ is the sub-complex of $D(X)$ defined by $DC_r(X) = f^{-1}((-\infty, r])$. Since $DC_r(X)$ is homotopy equivalent to X_r, we can compute topological information about X_r from $DC_r(X)$ at all scales $r \geqslant 0$. In particular, we are interested in persistent homology and cohomology.

Persistent (Co-)Homology: Applying homology and cohomology to a filtration of simplicial complexes, we obtain a sequence of induced linear maps on homology and cohomology respectively:

$$H_p(\mathbb{K}) : H_p(K_1) \to H_p(K_2) \to \ldots \to H_p(K_n)$$
$$H^p(\mathbb{K}) : H^p(K_1) \leftarrow H^p(K_2) \leftarrow \ldots \leftarrow H^p(K_n)$$

for each $p \in \{0, \ldots, d\}$ and where the maps on homology are induced by inclusion and we denote $f_p^{i,j} : H_p(K_i) \to H_p(K_j)$ for $i \leqslant j$ for the resulting compositions of maps. The maps on cohomology are induced by restriction of cochains and we denote the resulting linear maps by $g_{i,j}^p : H^p(K_j) \to H^p(K_i)$, for $i \leqslant j$. The p-th persistent homology group for $i \leqslant j$ is given by $H_p^{i,j} = im f_p^{i,j}$, so that non-trivial elements in $H_p^{i,j}$ correspond to homology classes born at or before index i and which survive until at least index j. The difference $j - i$ is called the *index persistence* of such a class. For us, $K_i = f^{-1}((-\infty, r_i])$, and $r_j - r_i$ is the *persistence of the class*. In fact, all the persistent homology groups can be computed by a decomposition of the persistence module into interval modules [19]. The p-th persistence diagram captures the information about the birth and death of p-th homology classes as the filtration value increases. It consists of multisets of points in the extended upper left quadrant. Each point (r_i, r_j) in the diagram corresponds to a homology class born at index i and surviving until index j. Points that lie far above the diagonal have a large persistence and are hence considered important features distinct from smaller scale features due to noise. An example is presented in Fig. 3. Classes born at index i and which do not die at the final filtration index n are called essential and are associated to points of the form (r_i, ∞) in the plane, extended formally to $(\mathbb{R}\cup\{\infty\})^2$. The persistent cohomology groups can be defined analogously by $H_{i,j}^p = im\, g_{i,j}^p$. In fact, persistent homology and cohomology are dual [19] with identical persistence diagrams. To compute a basis for the persistent (co-)homology groups, we first assume without loss of generality that the filtration $\mathbb{K}$ has been refined to a simplex-wise filtration, where $K_j = \bigcup_{i=1}^j \sigma_i$, so that $K_{j+1} = K_j \cup \{\sigma_{j+1}\}$ and we hence add a single simplex in each step of the filtration. Given such a simplex-wise filtration, several algorithms (see *e.g.* [20]) are available to compute a basis of the persistent (co-)homology groups. We shall use the standard left-to-right reduction algorithm [20] adapted for cohomology as described in [19] for this purpose. Fig. 3 illustrates an example of a filtration and an associated (co-)homology basis.

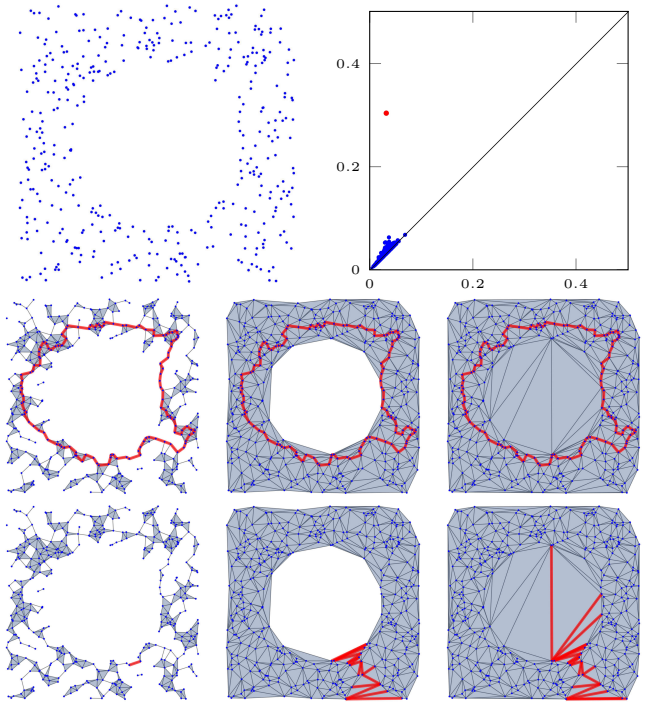

Fig. 3: Top row: A point-cloud $X \subset [0,1]^2$ and the 1^{st} persistence diagram of the Delaunay-Čech filtration $DC(X)$ with a single significant point $(0.031, 0.303)$ far above the diagonal and corresponding to the large hole in the middle of the point-cloud. Middle row: a 1-cycle c (over $\mathbb{Z}_2$) corresponding to the significant point and the complex at the filtration value 0.031 (left figure) when the cycle first corresponds to a non-trivial element $[c]$ in homology, at 0.150 (middle) and at the death filtration value of 0.303 (right) when the class $[c]$ becomes trivial in homology. Bottom row: a 1-cocycle φ (over $\mathbb{Z}_2$) dual to the previous 1-cycle at the same filtration intervals. Note how in the middle row, c is included into the subsequent complexes for higher filtration values, while in the bottom row, the cocycle is a result of restricting the co-cycle to smaller and smaller complexes.

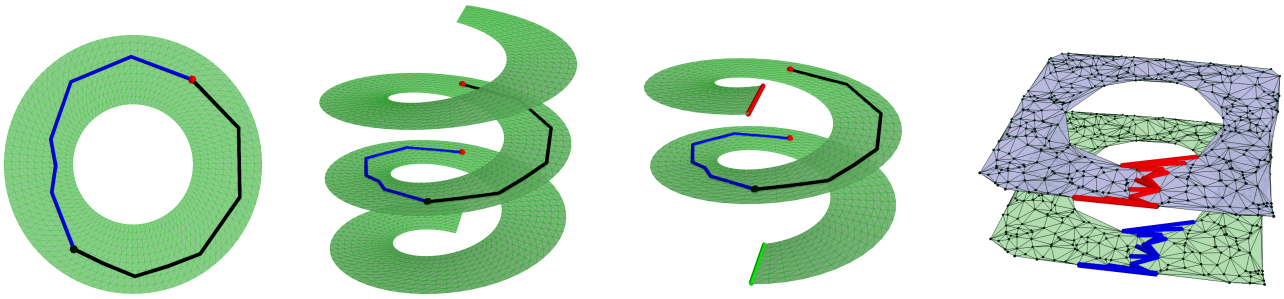

Fig. 4: Leftmost figure: A simple free configuration space $\mathcal{C}_f$ in the shape of an annulus and two paths in blue and black between the marked points. Center-left: A covering space of $\mathcal{C}_f$, where the z coordinate is given by the signed winding angle around the center of the annulus. The lifted black and blue paths now do not terminate in the same lifted coordinate and can hence be distinguished easily in the covering space. The covering space winds and extends infinitely far in both positive and negative z direction allowing us to classify trajectories between the two points in $\mathcal{C}_f$. Center-right: $\mathbb{Z}_2$ covering space of $\mathcal{C}_f$. Here, the red horizontal line is 'glued' to the green line, so that a trajectory winding twice around the hole in the center in $\mathcal{C}_f$ returns to the same z-coordinate also in the covering space. A figure similar to the left three plots appears in [18], where a generalization of these covering spaces is used for motion planning. Right: Example of our $\mathbb{Z}_2$ covering space which is built from noisy samples. It consists of 2 sheets of the simplicial complex $DC_r(X)$ which are glued along the indicated cohomology generator φ: Whenever the red line or blue line is crossed, we jump to the other layer in the covering space. Making two full rotations around the hole hence results in returning to the same z-coordinate layer as in the smooth $\mathbb{Z}_2$-covering space to the left. Unlike the previous approaches, the underlying covering space is generated fully from sampled data and we do not require a potentially imprecise numerical floating point integration to determine the analogue of the winding angle along a trajectory. The one-skeleton of this covering space precisely consists of the graph $G(DC_r(X), \Phi)$, $\Phi = (\varphi)$.

III. Methodology

Definition 3.1: Consider simplicial homology and cohomology over a finite field $\mathbb{F}$. We call a tuple $(\mathcal{K}, \Phi)$, where $\mathcal{K}$ is a simplicial complex, $\Phi = (\varphi_1, \ldots, \varphi_k)$, and $\varphi_i \in Z^1(\mathcal{K})$ are such that $[\varphi_1], \ldots, [\varphi_k] \in H^1(\mathcal{K})$ are linearly independent, a *cohomologically annotated simplicial complex*. For a 1-chain γ, we call $\Phi(\gamma) = (\varphi_1(\gamma), \ldots, \varphi_k(\gamma)) \in \mathbb{F}^k$ the *signature of* γ (a discrete analogue of the signature of [21]). The pairing induced by evaluation of 1-cochains on 1-chains yields a duality between $H_1(\mathcal{K})$ and $H^1(\mathcal{K})$ over field coefficients. In particular, for $c, c' \in Z_1(\mathcal{K})$, $\Phi(c) \neq \Phi(c')$ implies $[c] \neq [c'] \in H_1(\mathcal{K})$, and if $\Phi = (\varphi_1, \ldots, \varphi_k)$ yields a basis of $H^1(\mathcal{K})$, it is also true that, if $\Phi(c) = \Phi(c')$ then $[c] = [c'] \in H_1(\mathcal{K})$.

Lemma 3.2: Let γ, γ' be oriented paths of 1-simplices between 0-simplices, $s, t \in \mathcal{K}$, so that $\partial\gamma = t - s = \partial\gamma'$ and let $(\mathcal{K}, \Phi = (\varphi_1, \ldots, \varphi_k))$ be a cohomologically annotated simplicial complex. If $\Phi(\gamma) \neq \Phi(\gamma')$ then γ, γ' are not homotopy equivalent in $|\mathcal{K}|$.

Proof: Suppose $\Phi(\gamma) \neq \Phi(\gamma')$. Then there exists $j \in 1, \ldots, k$ such that $\varphi_j(\gamma) - \varphi_j(\gamma') = \varphi_j(\gamma - \gamma') \neq 0$, but then $[\gamma - \gamma'] \neq 0 \in H_1(\mathcal{K})$, since otherwise $\gamma - \gamma' = \partial c$ for some $c \in C_2(\mathcal{K})$, but then $\varphi_j(\partial c) = (\delta(\varphi_j))(c) = 0$, since $\varphi_j \in Z^1(\mathcal{K})$ which implies $\varphi_j(\gamma) - \varphi_j(\gamma') = 0$. Since $\gamma - \gamma'$ is non-trivial in $H_1(\mathcal{K})$ it is non-trivial in $\pi_1(\mathcal{K})$ (via Hurewicz' theorem [13]). Since $\gamma - \gamma'$ is non-trivial in $\pi_1(\mathcal{K})$, the paths γ and γ' are not homotopy equivalent. ∎

Definition 3.3: Denote by $\mathcal{K}^1$ the 1-skeleton of a cohomologically annotated simplicial complex $(\mathcal{K}, \Phi = (\varphi_1, \ldots, \varphi_k)$ with 0-simplex set V (vertices) and 1-simplex set E (edges). We define the covering graph corresponding to $(\mathcal{K}, \Phi)$ to be the finite directed graph $G = G(\mathcal{K}, \Phi)$ with vertex set $W = \{w = (v, \lambda) \in V \times \mathbb{F}^k : v \in V, \lambda \in \mathbb{F}^k\}$ and where an edge is inserted from (v, λ) to (v', λ') precisely when there exists a 1-simplex $e = [v, v'] \in E$ such that $\Phi(e) = \lambda' - \lambda$. For a vertex $w = (v, \lambda) \in G$, we denote by G_w the maximal connected component of G containing w.

Note that, for each arc $a = ((v, \lambda), (v', \lambda')) \in G$, there exists a corresponding edge $\pi(a) = (v, v')$ in $\mathcal{K}^1$, and we have the following result:

Corollary 3.4: Suppose there exist two sequences of directed arcs in $G(\mathcal{K}, \Phi)$ from $(v, 0)$ to (v', λ_1) and to (v', λ_2) respectively such that $\lambda_1 \neq \lambda_2$. Then the corresponding trajectories t_1, t_2 formed by sequences of edges of $\mathcal{K}^1$ under the projection $\pi : G \to \mathcal{K}^1$ are homotopy inequivalent.

Proof: We have $\Phi(t_1) = \lambda_1 \neq \lambda_2 = \Phi(t_2)$ and the result follows from the previous lemma. ∎

The search for homotopy inequivalent trajectories (up to homology) can hence be reformulated as a graph search for trajectories in $G(\mathcal{K}, \Phi)$. Note that, for $\mathbb{F} = \mathbb{Z}_p$, we are able to synthesize trajectories winding up to $p - 1$ times around any particular tunnel/void specified by each cohomology generator. In the case of $\mathbb{Z}_2$, which can be very efficiently implemented, we can in particular detect whether we pass a corresponding set of obstacles 'to the left or to the right' in 2D as well as its more complex generalization to higher dimensions.

Efficient Implicit Graph Representation and Search: Suppose that we are computing cohomology over the field $\mathbb{F} = \mathbb{Z}_p$ with p elements, where p is a prime. The covering graph $G(\mathcal{K}, \Phi)$, for $\Phi = (\varphi_1, \ldots, \varphi_k)$ and $\mathcal{K}$ with $|V|$ vertices and $|E|$ edges then has $|V|p^k$ vertices and significantly more edges than $\mathcal{K}^1$. These graphs quickly grow too large to fit into memory as k and p is increased. Our approach will hence be to run a graph search using an efficient implicit encoding of the graph. For this, we create an augmented graph $H = H(\mathcal{K}, \Phi)$ as follows: H has a node for each vertex in $\mathcal{K}^1$ and a directed edge (v, v') and (v', v) for each edge (v, v') in $\mathcal{K}^1$. All arcs are stored in an array $H.arcs$ and vertices in an array $H.vertices$. Each arc $a = (v, v')$ stores the index of its target vertex v' in $a.target$. We store the distance $a.dist = \|v - v'\|$ and the index of the next arc with the same source vertex $a.next\_out$ which is set to -1 if there are no further arcs. For each vertex v, we

Algorithm 1 Lifted Dijkstra

procedure LIFTEDDIJKSTRA($H(\mathcal{K}, \Phi = (\varphi_1, \dots, \varphi_k)), source\_id$)
  $N \leftarrow number\_of\_vertices(H)p^k$
  $d \leftarrow (\infty, ..., \infty) \in (\mathbb{R} \cup \{\infty\})^N$
  $f \leftarrow (false, ..., false) \in \mathbb{Z}_2^N$ // frontier vertices
  $s \leftarrow (false, ..., false) \in \mathbb{Z}_2^N$ // solved vertices
  $Q.insert(0, lift(source\_id, \mathbf{0}))$ // priority queue
  $f[lift(source\_id, \mathbf{0})] \leftarrow true$
  $d[lift(source\_id, \mathbf{0})] \leftarrow 0$
  while Q is not empty **do**
    $u\_id \leftarrow Q.extract\_min()$
    $s[u\_id] \leftarrow true$
    $f[u\_id] \leftarrow false$
    $arc\_id \leftarrow H.vertices[\pi_1(u\_id)].first\_out$
    while $arc\_id \neq -1$ **do**
      $a \leftarrow H.arcs[arc\_id]$
      $w\_id \leftarrow lift(a.target, \pi_2(u\_id) + a.mask)$
      if $s[w\_id] = false$ **then**
        $dist \leftarrow d[u\_id] + a.dist$
        if $dist < d[w\_id]$ **then**
          $d[w\_id] \leftarrow dist$
          if $!f[w\_id]$ **then**
            $Q.insert(dist, w\_id)$
            $f[w\_id] \leftarrow true$
      $arc\_id \leftarrow a.next\_out$
  return d

store the index of the first outgoing arc as $v.first\_out$. We furthermore store the value of the augmentation $\Phi(a) \in \mathbb{Z}_p^k$ for a directed arc $a = (v, v')$ in the edge data structure by encoding $\Phi(a) = (m_1, \dots, m_k) \in \mathbb{Z}_p^k$ as a single unsigned integer $a.mask$ using $code(m_1, \dots, m_k) = \sum_{i=1}^k m_i p^{i-1} \in \{0, \dots, p^k - 1\}$. In our implementation, $p = 2$ and we use a 16 bit integer for $a.mask$, allowing for $0 \leqslant k \leqslant 16$. Note that H efficiently encodes all required information to explore a connected component of the covering graph $G = G(\mathcal{K}, \Phi)$: If $\mathcal{K}^1$ has $|V|$ vertices then H has the same number of vertices while G has $|V|p^k$ vertices. Let us denote the vertices of H by $v_0, \dots, v_{|V|-1}$ and the vertices of G by $w_0, \dots, w_{|V|p^k-1}$, where we identify the vertex (v_i, λ) in G with $w_{lift(i,\lambda)}$, where $lift(i, \lambda) = i + |V|code(\lambda)$. For $j \in \{0, \dots, |V|p^k - 1\}$, we define $\pi_1(j) = j \mod |V| \in \{0, \dots, |V|-1\}$ and $\pi_2(j) = code^{-1}(\lfloor j/|V| \rfloor) \in \mathbb{Z}_p^k$. Then, if $w_j = (v_i, \lambda)$, we have $\pi_1(j) = i$ and $\pi_2(j) = \lambda$. Using this encoding, the outgoing edges from $(v_i, \lambda) \in G$ are precisely of the form $(v_j, \lambda + a.mask)$, where $a = (v_i, v_j)$ is an edge in H and we hence only need to store H in memory. Alg. 1 summarizes how this data structure can be used to perform Dijkstra's algorithm on G using H. We found that it was most efficient to use two boolean vectors $f, s \in \mathbb{Z}_2^n$ to keep track of the frontier and finalized vertex set in Dijkstra's algorithm. We also only fill the priority queue Q (implemented using the sequence heap of [22]), containing vertices ordered by distance to the source vertex v_{source_id}, during the execution of the algorithm instead of filling the queue at the initialization step. We observe that d, f, s still require $O(|V|p^k)$ memory, as does Q in the worst case. However, the constants are small and our representation of G by means of H significantly reduces the memory overhead of storing the underlying graph G. When the algorithm terminates, each vertex $w_j \in G$ reachable from $w_{lift(source\_id,\mathbf{0})} \in G$ satisfies $d[j] \neq \infty$ and $d[j]$ then denotes the distance of the shortest path between these vertices. We can then recover the shortest path in the usual manner recursively from the vector of distances d. Note that a similar approach is clearly also applicable to algorithms such as A* which we shall investigate in future work.

Summary of our approach: As outlined in Fig. 1, our motion planning approach consists of several steps: Given a point-cloud dataset $X \subset \mathcal{C}_f$, we construct a filtration $\mathbb{K}$ representing our data at all scales. In our experiments, this is done by considering filtrations such as $DC_r(X)$, and filtrations arising from sub- and superlevel sets of a function on X. We compute the persistent cohomology cocycle generators for the filtration and determine a fixed filtration paramter r resulting in a complex $\mathcal{K}_r$ in $\mathbb{K}$. The choice of r can be constrained by prior information about X, e.g. by imposing constraints on the Betti numbers, such as requiring that K_r contains a single connected component, or by picking r, e.g. to fall in the mid-point of the lifetime of the most persistent 1-cohomology generator (see also [23]). While persistent cohomology and homology are abstractly isomorphic, we require cohomology here since we utilize a basis of 1-cocycles, which intuitively form fences in the configuration space, enabling us to distinguish how tunnels in $\mathcal{C}_f$ are traversed by 'counting fence intersections' via the signature Φ which is constructed by selecting a subset of these 1-cocyles to apply our cohomological graph annotation and motion planning. The choice of subset determines which tunnels in $\mathcal{C}_f$ should be considered for topological motion planning and a choice of the k most persistent 1-cocycles alive at a filtration value allows us to consider only the k most important obstacles/voids at a given scale.

Technical differences and relationship with previous works: The graph $G(\mathcal{K}, \Phi)$ proposed in our work forms a discrete analogue of the covering space considered for topological motion planning in [8], [9], [10]. These covering spaces arose by annotating a trajectory's coordinates with a $\mathbb{R}^k$ valued integral of a collection of differential 1-forms. In 2D, this corresponded to annotating these trajectories by winding angles with respect to a collection of specified points interior to obstacles. Fig. 4 illustrates this idea. A similar illustration can also be found in the recent work [18], which used these winding coordinates modulo a prime p, corresponding to quotienting the spiral in Fig. 4 after p full rotations and gluing the ends together. Our approach is related to this idea since our finite field graph annotation approach also corresponds to a similar quotienting (Fig. 4), but is discrete ($\mathbb{F}$) and data-driven in nature. The right part of Fig. 4 contrasts our approach to [18] in the case of $\mathbb{Z}_2$ coefficients. A key difference to these previous works is that our approach does not require a numerical integration but utilizes a $\mathbb{F}$-valued signature instead which leads to an increased efficiency. Furthermore, our approach does not require a procedure to skeletonize obstacles, resulting in novel application domains such as *a first example of topological motion planning for a multijoint robot*. Our implicit graph representation in Alg. 1 constitutes a further difference and,

unlike in previous work, we can apply our approach both in the case where $\mathcal{C}_f$ is densely sampled and in a scenario where we are only given a sparse sample, e.g. from a collection of collision free trajectories. Unlike the works [8], [9], [10], [18], we introduce a persistent cohomology-based approach with several types of filtrations.

IV. Experimental Evaluation

Delaunay-Čech Filtrations in 2D: In our first experiment, we consider sufficiently dense sets of uniformly sampled collision free points $X \subset \mathcal{C}_f \subset \mathbb{R}^2$, so that $\mathcal{C}_f \simeq X_r$ for some $r > 0$. Here, $\mathcal{C}_f$ is constructed by selecting random subsets of K of the obstacles displayed in the top left of Fig. 5. If we know a priori what some of the Betti numbers of $\mathcal{C}_f$ are (e.g. number of connected components and number of obstacles), we can compute the persistence diagram and choose r to lie *e.g.* in the middle of the largest filtration interval satisfying these constraints [23]. If no prior information is available, one can choose r to lie in a large interval in which the Betti numbers remain constant. Real world information about sensor range can also be taken into account to further constrain the choice of r. To evaluate the performance of the proposed algorithm, we sample between $N = 10000$ and one million collision free points lying at least at distance 0.01 from any of the obstacles in the environment. This guarantees that for filtration values of $r < 0.01$ all edges in the skeleton of $DC_r(X)$ are themselves collision free. The top left of Fig. 5 illustrates a case where $r = 0.03 > 0.01$. In our experiments, we pick a filtration of $r = 3N^{-\frac{1}{2}}$ which empirically ensured that the resulting complex $DC_r(X)$ was connected and had the correct number of 'holes' corresponding to each obstacle. Furthermore, we then have $r < 0.01$ for $N \geqslant 90000$, guaranteeing global collision free solution trajectories in our experiments for sufficiently large sample sizes. The bottom part of the figure displays 64 overlaid solution trajectories and the first persistence diagram for a world with 6 obstacles. In Fig.6, we break down the computation times for N between 10000 and 100000. We observe that, as the graph size grows quickly with the number of obstacles, the graph search begins to dominate the total computation time for a large number of obstacles. Table II summarizes the total computation times as the number of samples and the number of obstacles varies. For each sample size less than 500000, we repeat the experiment 10 times (with random subsets of k obstacles) and report mean computation times and standard deviations. The table shows that our approach is very fast for small numbers of obstacles. Note also that Dijkstra's algorithm computes the full distance vector using which any trajectory from the source vertex to any other target vertex and topological class can then be obtained (in less than $0.5s$ for any query destination and sample size). Furthermore, for each fixed source and target vertex, we find 2^k topologically distinct solution trajectories when our world has k obstacles and for 1 million samples and 10 obstacles and a given source vertex, Dijksta's algorithm returns information about 1.024 billion solution trajectories as stored in the returned

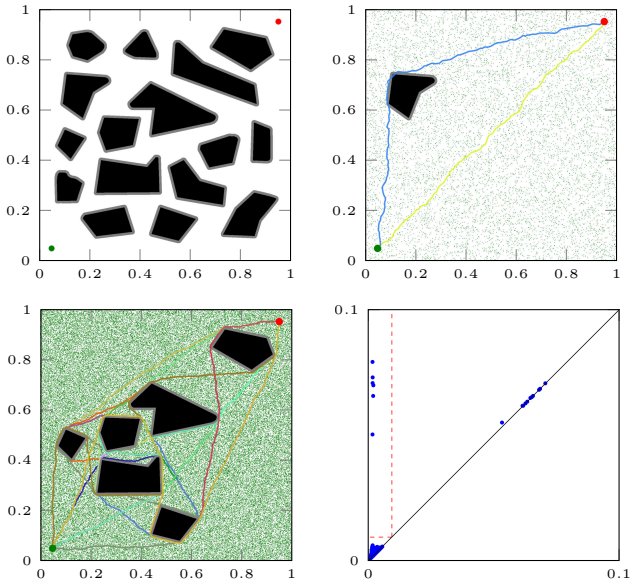

Fig. 5: Random subsets of the obstacles displayed in the top left (in black) are used in our experiments and we generate uniform random samples in the domain $[0,1]^2 \subset \mathbb{R}^2$ which are at least of distance 0.01 from each obstacle (outside the gray region). Initial and terminal positions are indicated as green and red points respectively. In the top right, we illustrate an example set of two solution paths in the case of a single obstacle, with 10000 samples (in green) and with a filtration value of 0.03. Note that, since $0.03 > 0.01$, $DC_{0.03}(X)$ can have edges that connect two collision free samples, but which are not themselves entirely collision free as illustrated in the top right part of this subfigure. For smaller filtration values, as in the bottom left, where we used 100000 samples and $r = 0.0095$, all edges are guaranteed to be collision free. $64 = 2^6$ homotopy inequivalent trajectories are shown overlayed onto each other. The bottom right diagram illustrates the associated first persistence diagram. The six points in the dashed region correspond to the obstacles and the dashed lines intersect at (r, r).

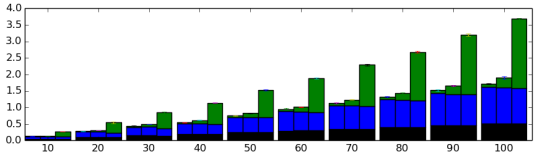

Fig. 6: Computation times (in s, vertical axis) for 2D experiments are broken into components: black: Delaunay triangulation computation in CGAL [24] and computation of the Delaunay-Čech filtration, blue: Computation of the persistent cohomology H^1 generators, green: graph construction and Dijkstra's algorithm. Each group of bars illustrates the time required (in seconds) for 0, 3, and 6 obstacles (left to right). The numbers of samples in thousands are shown along the horizontal axis.

distance vector in less than 12 minutes, and with $2^{10} = 1024$ homotopy inequivalent trajectories per source/target vertex pair. This in particular differs from the work in [11], where a smaller set of homotopy inequivalent trajectories (up to 10) was discovered incrementally in an *unbounded* search space.

PR2 Arm Motions: In this experiment, we simulate a PR2 robot using OpenRave [25]. The robot is placed next to a vertical bar obstacle as shown in red in Fig. 7 and we keep all but the 4 major arm joints of the left arm fixed. Our task is to determine trajectories of the left arm between the indicated initial and final configurations. For this purpose, we uniformly sampled 200000 collision free joint-configurations of the PR2's left arm. This results in a

Samples	Filtration	$DC_\infty(X)$		Number of obstacles										
		Edges	Triangles	0	1	2	3	4	5	6	7	8	9	10
10000	0.0300	29971.98 ± 3.49	19972.98 ± 3.49	0.1297 ± 0.0018	0.1311 ± 0.0014	0.1352 ± 0.0020	0.1427 ± 0.0014	0.1603 ± 0.0020	0.1978 ± 0.0022	0.2786 ± 0.0028	0.4395 ± 0.0026	0.6848 ± 0.0250	1.2819 ± 0.0036	2.5497 ± 0.0129
20000	0.0212	59970.50 ± 3.38	39971.50 ± 3.38	0.2769 ± 0.0148	0.2857 ± 0.0059	0.2956 ± 0.0048	0.3133 ± 0.0064	0.3539 ± 0.0062	0.4351 ± 0.0065	0.5647 ± 0.0356	0.8391 ± 0.0070	1.4587 ± 0.0108	2.7556 ± 0.0198	5.5616 ± 0.0425
30000	0.0173	89969.42 ± 3.63	59970.42 ± 3.63	0.4286 ± 0.0298	0.4530 ± 0.0032	0.4673 ± 0.0074	0.4954 ± 0.0072	0.5518 ± 0.0227	0.6034 ± 0.0060	0.8364 ± 0.0057	1.3104 ± 0.0170	2.2778 ± 0.0169	4.3247 ± 0.0383	8.8125 ± 0.0802
40000	0.0150	119968.88 ± 3.53	79969.88 ± 3.53	0.5545 ± 0.0098	0.5581 ± 0.0049	0.5747 ± 0.0054	0.6111 ± 0.0081	0.6833 ± 0.0070	0.8380 ± 0.0098	1.1396 ± 0.0161	1.7841 ± 0.0207	3.1182 ± 0.0564	6.0961 ± 0.1002	12.5567 ± 0.1999
50000	0.0134	149968.55 ± 3.64	99969.55 ± 3.64	0.7559 ± 0.0082	0.7676 ± 0.0116	0.7827 ± 0.0113	0.8315 ± 0.0125	0.9315 ± 0.0119	1.1254 ± 0.0094	1.5342 ± 0.0098	2.3750 ± 0.0192	4.2110 ± 0.0396	8.3736 ± 0.1390	17.2210 ± 0.1442
60000	0.0122	179967.89 ± 3.57	119968.89 ± 3.57	0.9425 ± 0.0091	0.9490 ± 0.0133	0.9805 ± 0.0125	1.0299 ± 0.0205	1.1488 ± 0.0113	1.3957 ± 0.0229	1.8948 ± 0.0145	2.9609 ± 0.0153	5.3004 ± 0.0548	10.6218 ± 0.1129	22.3609 ± 0.2281
70000	0.0113	209967.09 ± 3.60	139968.09 ± 3.60	1.1292 ± 0.0204	1.1448 ± 0.0136	1.1756 ± 0.0176	1.2429 ± 0.0272	1.3777 ± 0.0187	1.6838 ± 0.0227	2.2823 ± 0.0172	3.6670 ± 0.0324	6.6847 ± 0.1007	13.3552 ± 0.0845	28.2221 ± 0.2014
80000	0.0106	239966.45 ± 3.60	159967.45 ± 3.60	1.3227 ± 0.0187	1.3321 ± 0.0243	1.3800 ± 0.0159	1.4470 ± 0.0235	1.6081 ± 0.0156	1.9603 ± 0.0153	2.6818 ± 0.0278	4.3383 ± 0.0385	8.0440 ± 0.1235	16.1658 ± 0.1101	33.6077 ± 0.3051
90000	0.0100	269966.75 ± 3.79	179967.75 ± 3.79	1.5387 ± 0.0514	1.5475 ± 0.0344	1.5833 ± 0.0197	1.6919 ± 0.0165	1.8799 ± 0.0164	2.2838 ± 0.0156	3.1904 ± 0.0317	5.2207 ± 0.0629	9.7655 ± 0.1574	19.5922 ± 0.1564	40.1525 ± 0.2844
100000	0.0095	299966.78 ± 3.46	199967.78 ± 3.46	1.7163 ± 0.0190	1.7443 ± 0.0333	1.7917 ± 0.0232	1.9048 ± 0.0350	2.1205 ± 0.0113	2.5949 ± 0.0148	3.6863 ± 0.0306	6.0790 ± 0.0795	11.3504 ± 0.1101	22.8980 ± 0.2842	46.5861 ± 0.3633
500000	0.0042	1499960.55 ± 4.85	999961.55 ± 4.85	11.2779	11.5662	12.1993	13.0111	14.4668	18.2660	26.4173	44.8978	85.7563	160.3689	329.8708
1000000	0.0030	2999963.00 ± 4.26	1999964.00 ± 4.26	25.1642	25.4774	26.4277	28.7711	32.3689	40.2058	58.0553	98.7767	181.7561	361.8256	696.8684

TABLE II: Summary of 2-dimensional topological motion planning experiments. We report the total time (in seconds) required to compute all data structures, persistent homology and to execute the lifted Dijkstra algorithm for a fixed source vertex. For up to 100000 samples, each experimental setting is repeated 10 times with a random subset of k obstacles. For larger sample sizes, we report timing results of a single trial. Edge and triangle numbers are averaged over all trials and number of obstacle settings. We also report standard deviations where applicable.

simplicial 2-skeleton (we do not require higher dimensional simplices to compute H^1) filtration with 3675821 edges and 12703304 triangles in 4 dimensions which models the space X_r of unions of balls of radius r around the samples at all scales $r \geqslant 0$. Note that the resulting data structure yields a highly detailed model of the 4 dimensional free configuration space. The computation of the Delaunay triangulation took $37.6s$, the 2-skeleton extraction $32.2s$ and the Delaunay-Čech filtration computation took $8.38s$. The resulting first persistence diagram is displayed in the top left part of Fig.7 and was computed in $159.8s$. We can clearly see the red point indicating a particularly large persistence interval. At filtration $r = 0.11$, the simplicial complex is path-connected and $H^1(DC_r(X))$ has dimension 4716 (equal to the number of points in the top right area bordered by the dashed lines intersecting at (r, r)). We chose the cohomology generator corresponding to the red most persistent interval among these as our cohomological annotation and computed shortest trajectories between the indicated initial and terminal arm configuration. The graph construction and search took $2.1s$. We find two topologically inequivalent shortest paths illustrated in the second and third row of the figure respectively. The difference between these trajectories can be observed by considering the second picture in each column. The PR2's left upper arm is lowered in the first trajectory to pass the obstacle while the upper arm is extended upwards in the second solution trajectory. These two trajectories are clearly not homotopy equivalent – continuously deforming one into the other would require a horizontal posture of the upper arm resulting in a collision with the red obstacle. The selected large persistence interval corresponds to a cohomology class which is generated by a thick tunnel/void in the 4D configuration space corresponding to collisions of the PR2's left elbow with the obstacle. The chosen 1-cocycle yields a 'fence' which is pierced by the first solution trajectory but not by the second. *While this example may be considered simple, no previously known method is – to the best of our knowledge – currently able to synthesize such types of homotopy-inequivalent solution trajectories in an automated sampling-driven manner.* Our method hence makes available a new wealth of trajectory classes. In case one of the trajectories becomes obstructed due to changing environment conditions, the robot could for example fall back to the previously computed second trajectory solution.

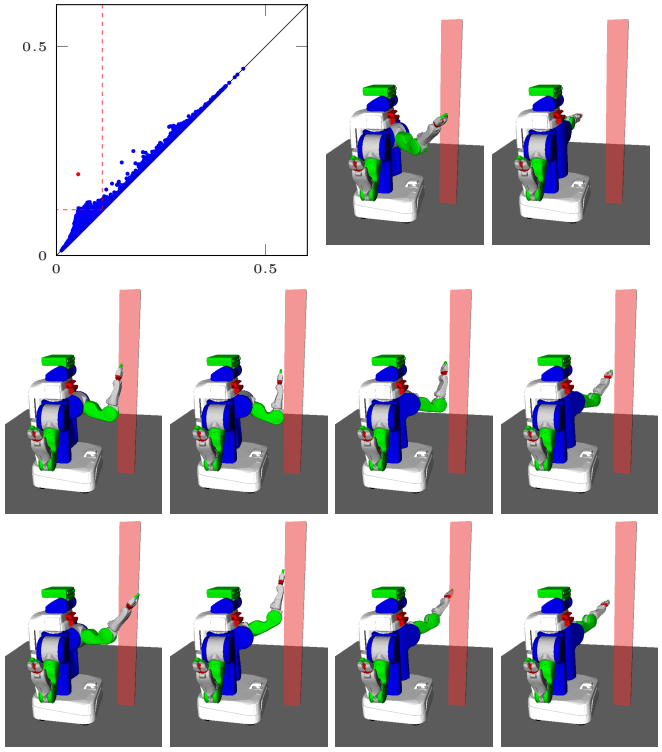

Fig. 7: Top-left: first persistence diagram for Delaunay-Čech filtration of 200000 uniform collision free joint-configuration samples X of the left arm in $\mathbb{R}^4$. At selected filtration value of $r = 0.11$, $DC_r(X)$ has one connected component and $dim(H^1(DC_r(X))) = 4716$ (number of points in the upper left dashed region), with a persistence interval with particularly large persistence marked in red.

Note also that any continuous optimization of a trajectory in each homotopy class will have to remain within its initial homotopy class. Our trajectory solutions can hence be used to provide initializations to optimizers such as CHOMP [26].

Trajectory datasets: Here, we study applications of our approach in the case where we are given point-clouds consisting of real-world trajectories. Unlike before, such a point-cloud X does not necessarily represent a dense sample of $\mathcal{C}_f$. Instead, the intrinsic shape of X_r gives rise to holes which might be due to obstacles or parts of space not covered by X.

Racecar dataset: The first trajectory dataset consists of 3324 GPS data points $X \subset \mathbb{R}^2$ of a racecar driving around a racetrack [27] and is displayed in the top left of Fig. 8, with its first persistence diagram displayed to

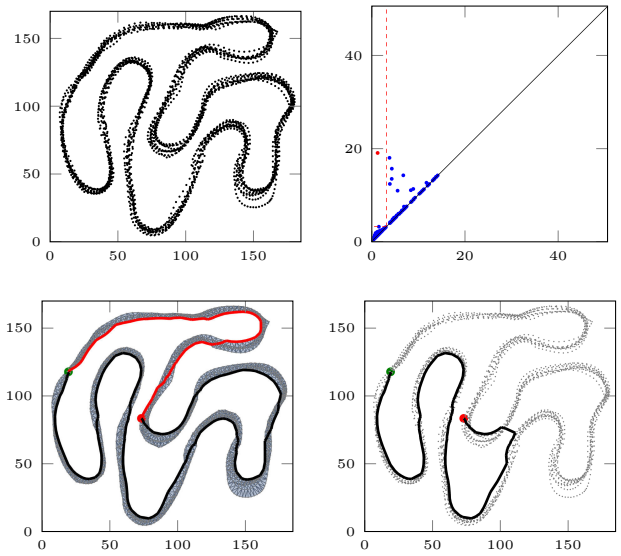

Fig. 8: Top: Racecar GPS point-cloud and first persistence diagram with $r = 3.276$ marked by dashed lines. Bottom left: $DC_r(R)$ and overlaid found shortest paths at $r = 3.276$. Bottom right: a solution trajectory for $r = 4.0$.

the right. By investigating the 0^{th} persistence diagram, we find that $X_r \simeq DC_r(X)$ is connected for $r \geqslant 2.26$. For $r = 3.276$, $dim(H^1(X_r)) = 1$ and we obtain the gray simplicial complex in the bottom left. We find two solution trajectories (in red and black) between the indicated points which generalize the recorded trajectories (all computed in less than $0.5s$). When we increase the filtration value to $r = 4$, $dim(H^1(X_r)) = 2$ and we obtain 4 solution trajectories. An example such trajectory is displayed in the bottom right of the figure. Note how this trajectory represents a reasonable solution generalizing the recorded trajectory behavior if we allow ourselves to trust that samples of distance less than or equal to 4 will be path-connected in C_f.

Motion capture dataset: We recorded the set of 3D trajectories displayed in Fig. 9 using a Nest of Birds motion capture rig. The trajectory set consists of two motion primitives displayed in blue and red respectively and contains 8998 data-points $X \subset \mathbb{R}^3$. In this experiment, we illustrate how our topological motion planning approach enables us to synthesize new motions which combine the behavior of these two classes. The two voids in this data-set are clearly visible as two points with large persistence in the first persistence diagram in the top right of Fig. 9. The dashed lines intersect at $(0.085, 0.085)$. We computed the corresponding 4 homotopy inequivalent solution trajectories displayed in the bottom left of the figure (total computation time: less than $6s$). Note that the yellow and blue trajectories correspond to an optimized shortest path in the red and blue motion primitive respectively. Additionally, we are able to recover novel motion behavior mixing the two primitives as indicated by the red and black trajectory which traverse both voids either to the left or to the right. We believe that an investigation of our approach together with large motion capture/gesture databases might be of interest, especially since $DC_r(X)$ and persistent cohomology generators can in that case be computed offline, reducing the online computation

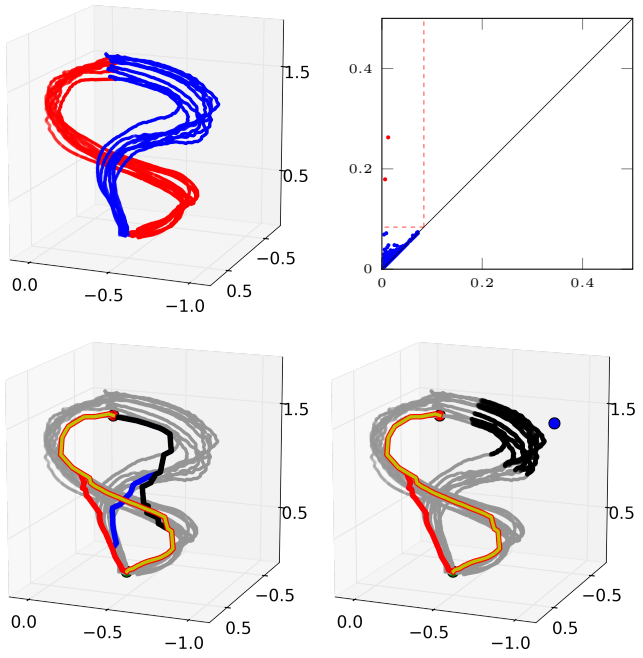

Fig. 9: Top left: 3D motion capture data of two motion primitives. Top right: associated first persistence diagram with marked filtration value $r = 0.085$. Bottom left: Four shortest homotopy inequivalent solution trajectories at this filtration value are displayed on top of the gray data points. Bottom right: A cost function defined by the distance to the blue threat/obstacle yields a sub-filtration with only two solution trajectories at $r = 0.78$.

cost of synthesizing new trajectories to just graph-search.

The bottom right part of Fig. 9 illustrates how a cost function can be combined with our approach (see also [12]). Observing that, for $r = 0.085$, $DC_r(X)$ provides a good model for our point-cloud, consider now the emergence of an agent indicated by the blue dot in the figure. To adapt our trajectory planning approach, we can define a cost function $c : \mathbb{R}^3 \to \mathbb{R}$, e.g. by $c(x) = exp(-d(a, x)^2)$ and a new filtration F_r of the complex $DC_r(X)$ where a k-simplex $[v_0, \ldots, v_k]$ has filtration value $\max_{i \in 0,\ldots,k} c(v_i)$. For $r = 0.78$, F_r does not contain any simplex involving any of the black vertices, allowing for only the two indicated solution trajectories, while for large filtrations, we recover all the solutions shown in the bottom left. We believe that the investigation of such data-driven sub-filtrations expressing multiscale constraints arising from cost, likelihood and navigability functions provides an interesting direction for future work.

V. Conclusion

In this work, we have introduced the use of persistent cohomology with finite field coefficients as a tool for data-driven homotopy-aware motion planning. Our experiments show that our technique can be used with large sample sizes in low dimensions and we have demonstrated that we can plan topologically distinct trajectories in application domains in which topological motion planning techniques were previously not applicable. In future work, we intend to investigate the combination of our methods with probabilistic and cost based filtrations as well as the use of alternatives to the underlying Delaunay-Čech complexes.

REFERENCES

[1] S. Thrun, W. Burgard, and D. Fox, "Probabilistic robotics," 2005.

[2] S. M. LaValle and J. J. Kuffner, "Rapidly-Exploring Random Trees: Progress and Prospects," in *Algorithmic and Computational Robotics: New Directions*, B. R. Donald, K. M. Lynch, and D. Rus, Eds. Wellesley, MA: A K Peters, 2001, pp. 293–308.

[3] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[4] J. T. Schwartz and M. Sharir, "On the piano movers problem. II. General techniques for computing topological properties of real algebraic manifolds," *Advances in Applied Mathematics*, vol. 4, no. 3, pp. 298–351, 1983.

[5] J. Canny, *The complexity of robot motion planning*. MIT press, 1988.

[6] J.-C. Latombe, *Robot Motion Planning*. Springer, 1991.

[7] S. Bhattacharya, V. Kumar, and M. Likhachev, "Search-based path planning with homotopy class constraints," in *Proc. of The Twenty-Fourth AAAI Conf. on Artificial Intelligence*, 11-15 July 2010.

[8] S. Kim, K. Sreenath, S. Bhattacharya, and V. Kumar, "Optimal trajectory generation under homology class constraints," in *51st IEEE Conf. on Decision and Control*, 10-13 Dec 2012.

[9] S. Bhattacharya, M. Likhachev, and V. Kumar, "Identification and representation of homotopy classes of trajectories for search-based path planning in 3D," in *Proc. of Robotics: Science and Systems*, 27-30 June 2011.

[10] S. Bhattacharya, D. Lipsky, R. Ghrist, and V. Kumar, "Invariants for homology classes with application to optimal search and planning problem in robotics," *Annals of Mathematics and Artificial Intelligence (AMAI)*, April 2013.

[11] S. Bhattacharya, M. Likhachev, and V. Kumar, "Topological constraints in search-based robot path planning," *Autonomous Robots*, vol. 33, no. 3, pp. 273–290, October 2012.

[12] F. T. Pokorny, M. Hawasly, and S. Ramamoorthy, "Multiscale topological trajectory classification with persistent homology," in *Proceedings of Robotics: Science and Systems*, July 2014.

[13] A. Hatcher, *Algebraic topology*. Cambridge University Press, 2002.

[14] H. Edelsbrunner and J. L. Harer, *Computational topology: an introduction*. AMS Bookstore, 2010.

[15] H. Edelsbrunner and J. Harer, "Persistent homology-a survey," *Contemporary mathematics*, vol. 453, pp. 257–282, 2008.

[16] G. Carlsson, "Topology and data," *Bull. Amer. Math. Soc. (N.S.)*, vol. 46, no. 2, pp. 255–308, 2009.

[17] U. Bauer and H. Edelsbrunner, "The Morse theory of Čech and Delaunay filtrations," in *Proc. of the Thirtieth Annual Symp. on Comp. Geometry*, ser. SOCG'14. New York, NY, USA: ACM, 2014, pp. 484:484–484:490.

[18] S. Bhattacharya, R. Ghrist, and V. Kumar, "Persistent homology for path planning in uncertain environments," *Preprint*, June 2014.

[19] V. De Silva, D. Morozov, and M. Vejdemo-Johansson, "Dualities in persistent (co) homology," *Inverse Problems*, vol. 27, no. 12, p. 124003, 2011.

[20] U. Bauer, M. Kerber, and J. Reininghaus, "PHAT (Persistent Homology Algorithm Toolbox)," http://code.google.com/p/phat/.

[21] S. Bhattacharya, M. Likhachev, and V. Kumar, "Topological constraints in search-based robot path planning," *Autonomous Robots*, vol. 33, no. 3, pp. 273–290, October 2012, dOI: 10.1007/s10514-012-9304-1.

[22] P. Sanders, "Fast priority queues for cached memory," *ACM Journal of Experimental Algorithmics*, vol. 5, pp. 312–327, 1999.

[23] F. T. Pokorny, C. H. Ek, H. Kjellström, and D. Kragic, "Persistent homology for learning densities with bounded support," in *Advances in Neural Information Processing Systems 25*, P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., 2012, pp. 1826–1834.

[24] "CGAL, Computational Geometry Algorithms Library," http://www.cgal.org.

[25] R. Diankov and J. Kuffner, "OpenRAVE: A Planning Architecture for Autonomous Robotics," Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34, Jul. 2008.

[26] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 489–494.

[27] KTH Racing, Formula Student Team, KTH Royal Institute of Technology, Stockholm, Sweden, 2012.