Path Clustering with Homology Area

J. Frederico Carvalho[1], Mikael Vejdemo-Johansson[2], Danica Kragic[1] and Florian T. Pokorny[1]

***Abstract*— Path classification has found many applications in recent years. Common approaches to this problem use aggregates of the distances between points to provide a measure of dissimilarity between paths which do not satisfy the triangle inequality. Therefore, such distance measures need to be used with care to avoid cluster chaining. Furthermore, they do not take into account the topology of the space where the paths are embedded. To tackle this, we extend previous work in path clustering with homology, by employing homology area as a measure of distance between homologous paths in a triangulated mesh. Further, we show that the resulting distance satisfies the triangle inequality, and how we can exploit the properties of homology to reduce the amount of pairwise distance calculations necessary to cluster a set of paths by an order of magnitude.**

I. Introduction

Due to the prevalence of devices with access to positions sensors such as GPS, large datasets of paths have become increasingly available in the past few years [1]. Through path clustering, this data can be partitioned into sets of similar paths, from which motion primitives can be learned in order to control a robot without explicitly programming it [2], [3]. In this paper we present a new method of computing a topologically inspired similarity measure for paths which can be applied to path clustering.

Clustering is a machine learning technique used to classify large sets of data into smaller subsets that are similar to each other [4]. However, the domain of "popular" clustering techniques, such as k–means and support vector machines, is that of data points in a fixed dimension vector space. Because paths have variable length, they cannot be easily represented in finite dimensions. Therefore, commonly used path clustering methods rely on calculating distances between paths and applying clustering methods that use distance alone [5].

These methods have been successfully applied to different problems, such as learning dynamic scene models, and car trajectories [6]–[8] to name a few. Comparative surveys of commonly used path distances can be consulted in [9], [10]. The aforementioned distances however, do not take into account the topology of the space the paths are drawn from, and therefore can not properly gauge the influence of obstacles on the clustering produced [11]. To tackle this, in [11] we proposed a method for path clustering that only takes the obstacles into account.

In this paper we propose an extension of the framework of clustering with relative homology we proposed in [11], by using homology area [12]. Intuitively, this corresponds to defining the distance between two paths as the area enclosed between them (see an example in Figure 1). This method allows us to obtain a more fine-grained control of how the clustering takes place over only using relative homology. It also ensures that the obtained distance function is compatible with the results from homology clustering.

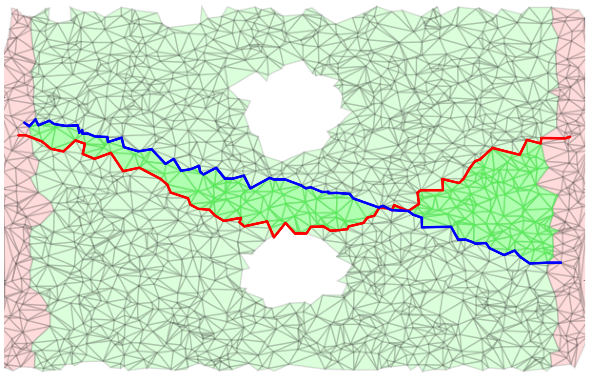
Fig. 1: Triangulated region of the plane X (green and red) with two holes (white). The region in red corresponds to a subset $A \subseteq X$ that are used as "goal regions" which contains the end points of the paths being analyzed. We also depict two edge-paths p, q (in red and blue) together with the minimum area bounding chain c in bright-green, which intuitively corresponds to the area that is delimited by the paths p and q. The area of the region c is interpreted as the distance between paths p and q.

Further, we show how this path distance, even though computationally expensive to compute, can be leveraged when calculating a distance array to cluster a large set of paths, outperforming DTW[1] by an order of magnitude.

More details of our method can be consulted in `http://csc.kth.se/~jfpbdc/projects/homology_clustering`.

A. Motivation and Related Work

The problem of learning by demonstration deals with extracting motion primitives for a robot from data collected from a human demonstration, rather than explicit programming [2]. However if motion primitives are averaged from a diverse enough set of trajectories, the resulting motion may not be a good representative[2] of the demonstrated motions. Therefore path clustering can be used to split the space of paths into cohesive sets of paths from which valid motion primitives can be extracted [3].

[1] CAS/RPL, KTH, Royal Institute of Technology, Stocholm, Sweden. `{jfpbdc,dani,ftpokorny}@kth.se`
[2] Mathematics Department, CUNY College of Staten Island, New York, USA. `Mikael.VejdemoJohansson@csi.cuny.edu`

[1]Dynamic Time Warping (DTW) is a popular path distance which can be efficiently computed [13].
[2]If the distribution of paths is multimodal, then taking an average of the motions may yield something that lies outside of the scope of the data.

Path clustering has been extensively researched since the early 2000s and has found applications in surveillance, traffic analysis, among others [6]–[9], [14]. In order to cope with the high dimensionality of the space of paths on a vector space, the employed methods use distance-based clustering rather than other methods that require a finite dimensional representation, such as k–means and support vector machines. An exhaustive review of path clustering combining different distances and clustering methods can be consulted in [15]. As mentioned before, the employed clustering methods rely only on distance between paths, to assign them to a cluster, therefore they need to compare each path to every other. This implies that the path distance needs to be evaluated $O(N^2)$ times where N is the number of paths. This fact puts a strain on path clustering methods, since path distances need to be evaluated fast in order to cope with large databases of paths.

Recently in [11] we proposed a clustering method for paths based solely on their homology class. Intuitively, two paths belong to the same cluster (in which case they are said to be "homologous") if there are no obstacles in the area between them. The strength of this method, lies not only on the speed with witch it can be evaluated, but also on the fact that it only needs to do a single pass through the data, making it significantly outperform other methods. However, since this method only distinguishes paths based on topological features, the paths in each cluster present a wide range of variability, we therefore want to be able to cluster paths into more narrow categories in a manner that is compatible with the homology clusters. To this end we use the results from [12] where we presented an efficient method for computing the area between two closed paths as the region whose boundary is the union of the two paths. This notion of distance has the convenience of being compatible with the purely topological clustering in [11].

In this paper we build upon the results in [11], [12] by extending the notion of minimum area homology to quotient complexes, and employing the area of the obtained region as a path distance. Further, we note that in clustering a set of N paths with k homology classes, the number of regions that need to be computed can be brought down from $O(N^2)$ to $O(kN)$ in the worst case. We show in the experiments section how this allows us to outperform DTW [13] when clustering even relatively small datasets (circa 2000 paths).

B. Mathematical Background

We now provide a brief description of the mathematical tools used in the remainder of the paper (an exhaustive treatment can be found in [16]). To do this we begin by introducing Homology, which is a general algebraic tool that can be applied to understanding topological spaces, particularly through Simplicial Homology which we explain further on. Finally, we introduce relative homology and homology area which will use to define our path distance.

1) Homology: A chain complex is a pair $(C_\bullet, \partial_\bullet)$, where $C_\bullet = \{C_i\}_{i\in\mathbb{N}}$ is a sequence of vector spaces, and $\partial_\bullet = \{\partial_i : C_i \to C_{i-1}\}_{i\in\mathbb{N}}$ is a sequence of linear maps (matrices) called *boundary maps* (matrices) often depicted in a sequence as in (1).

$$\cdots \leftarrow C_i \xleftarrow{\partial_i} C_i \xleftarrow{\partial_{i+1}} C_{i+1} \leftarrow \cdots \tag{1}$$

Where the boundary matrices are required to satisfy $\partial_i\partial_{i+1} = 0$. We call an element of C_i an *i–chain*; and when no confusion arises, we denote all boundary maps as ∂, dropping the index.

Given a chain complex, we define two further sequences of vector spaces $Z_\bullet, B_\bullet$ by:

$$Z_i = \ker(\partial_i) \quad B_i = \operatorname{im}(\partial_{i+1})$$

We call elements of Z_i and B_i; *i–cycles* and *i–boundaries*, respectively. Further, since $\partial_i\partial_{i+1} = 0$ implies $B_i \subseteq Z_i$, we can define the quotient $H_i = Z_i/B_i$, which we call the i–th *homology* of the chain complex.

Given an i–boundary p and an $(i+1)$–chain c such that $p = \partial c$ we call c the *bounding chain* of p.

2) Simplicial Homology: Given $k+1$ affinely independent points in a vector space $\{x_0, \ldots, x_k\} \subseteq \mathbb{R}^n$ with $n > k$, we form the *k–simplex* $[x_{0:k}]$ (where $x_{0:k}$ is a shorthand for "$x_0, \ldots, x_k$"). Each such simplex has an orientation that is inherent to the ordering of its points or *vertices*. Two simplices with the same vertices are said to have opposite orientations if the ordering differs by an odd permutation.

Geometrically, we represent a simplex by the convex hull of its vertices, and its boundary by

$$\operatorname{bd}([x_{0:k}]) = \bigcup_{i=0}^{k} [x_{0:i-1}, x_{i+1:k}]$$

Given a topological space $X \subseteq \mathbb{R}^n$, we can obtain a discrete approximation of X by taking a discrete set $\tilde{X} \subseteq X$ and using a triangulation T of $\tilde{X}$ (in the remaining we let T be a subset of the Delaunay triangulation). From T, we can build a chain complex as follows:

In order to construct a chain complex out of $\tilde{X}$ and T, we denote the ordered set of i–simplices in T by $T^{(i)} = \{\sigma_{1:b_i}\}$, and define $C_i = \mathbb{R}^{b_i}$. Furthermore, for each $j = 1, \ldots, b_i$ we set

$$\partial_i e_j = \sum_{k=0}^{i} \pm e'_{l_{i,k}}$$

where $e_{1:b_i}$ and $e'_{1:b_{i-1}}$ form the canonical basis of C_i and C_{i-1}, respectively; and $l_{i,k}$ is the index of the k–th face of σ_j and the sign depends on the orientation.

Thus, we obtain a *simplicial complex* (see [16] for a proof), which we denote by $(C_\bullet(X), \partial_\bullet)$ and its homology by $H_\bullet(X)$.

In this setting, it can be observed that 1–cycles correspond to sums of edges that form a closed path, and that 1–boundaries correspond to those 1–cycles that form the boundary of a closed region (see Figure 2 for an example).

3) Relative Homology: Given a chain complex $(C_\bullet, \partial_\bullet)$ we define a *subcomplex* as a complex $(A_\bullet, \partial_\bullet)$ where for each i, A_i is a subspace of C_i and ∂_i is the boundary map of $(C_\bullet, \partial_\bullet)$ restricted to A_i. From such a subcomplex we can

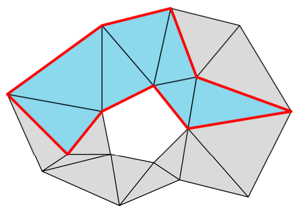
Fig. 2 a: A cycle which is a boundary (in red) and its bounding chain (in blue).

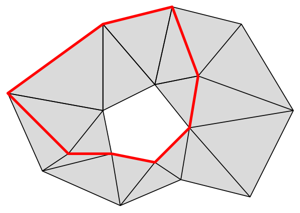
Fig. 2 b: A cycle which is not a boundary (in red).

define *relative chain complex* $(D_\bullet, \partial'_\bullet)$, as $D_i = C_i/A_i$ and ∂'_i as the map induced by ∂_i on the quotient.

In the case of simplicial complexes, as introduced in Section I-B.2, we are interested in subcomplexes $A_\bullet$ such that the generators of each A_i are a subset of the generators of C_i. This ensures that $A_\bullet$ can be seen as a closed subspace $A \subseteq X$. In this case we denote the relative chain complex by $(C_\bullet(X, A), \partial_\bullet)$ and its homology by $H_\bullet(X, A)$.

Now, taking the quotient $C_i(X)/C_i(A)$ corresponds to ignoring all topological features in A, and hence we expect to see a relationship between the relative complex and the quotient X/A (see Figure 3) which is made explicit in the following Lemma:

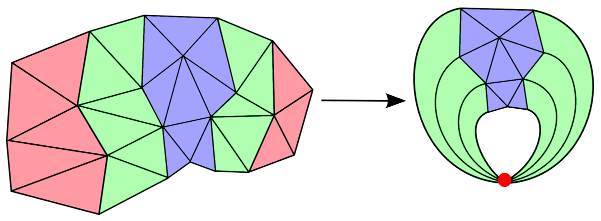
Fig. 3: Representation of the quotient map on a simplicial complex. The region in blue is unaffected whereas the region in red is collapsed onto a single point and the region in green is distorted. In this example, we have taken a space with no holes and have produced a space with one hole.

Lemma 1 ([16]). *Let X be a simplicial complex and A be a subcomplex, then the relative homology $H_i(X, A)$ is isomorphic to $H_i(X/A)$*

4) Homology area: We say that a simplicial complex is k–dimensional if it has no simplices of dimension higher than k. Let S be a two-dimensional simplicial complex, and let $S^{(2)} = \{\sigma_{1:b_2}\}$, then for any 2–chain $c = (c_{1:b_2}) \in C_2(X)$ as

$$\text{area}(c) = \sum_i |c_i| \text{Area}(\sigma_i)$$

where $\text{Area}(\sigma)$ denote the area of the 2–dimensional simplex σ. Finally, we define the *minimum area bounding chain*[3] of a 1–cycle p as

$$c^* = \operatorname*{arg\,min}_{c \in C_2(X)} \ \text{area}(c) \qquad \text{subject to: } \partial c = p \qquad (2)$$

[3]In [12] we call it a “minimum area homology”.

II. Methodology

We start this section by presenting and describing the algorithms that make up the main contributions of this paper as well as provide the complexity of Algorithm 2. In the remainder of the section we introduce the theory that ensures the correctness of our algorithms.

A. Algorithms

We now present the algorithm to compute the quotient of a simplicial complex (Algorithm 1), and to compute the homology-area of associated to a set of paths (Algorithm 2).

Algorithm 1: Computing a quotient where the support of a characteristic function is collapsed to a single point

```
input  : $S$ /* simplicial complex */
input  : $q$ /* characteristic function */
/* $q$ satisfies $q(x) = 1$ for $x \in A$, */
/* and $q(x) = 0$ otherwise. */
/* $S = (pts, tris)$ where: */
/* $pts$: list of 2D points. */
/* $tris$: list of triangles, each is */
/* a triple of indices in $pts$. */
output : $Q$ /* simplicial complex */
/* $Q$ is a representative of $S/A$ */
1  $qPts = \{Null\}$;
   /* 1st is the image of $A$ */
2  for $p \in pts$ do
3      if $q(p) = 0$ then
4          append $p$ to $qPts$;
5  $qTris = \{\}$;
6  for $tri \in tris$ do
7      $qTri = \{\}$;
8      $has0 = False$;
9      for $v \in tri$ do
10         if $q(pts[v]) = 1$ then
11             $has0 = True$;
12         else
13             $qv$ = index of $v$ in $qPts$;
14             append $qv$ to $qTri$;
15     if $has0$ then
           /* include 0 at most once */
16         prepend 0 to $qTri$;
17 return $Q = (qPts, qTris)$
```

In Algorithm 1 we assume the simplicial complex S is given as a pair $(points, tris)$ where $points \subseteq \mathbb{R}^2$ is a (finite) set containing the vertices of the simplicial complex, and $tris$ is a triangulation of those points, i.e. $tris = S^{(2)}$. Similarly, the quotient Q is represented by a pair of points and triangles $(qPts, qTris)$.

The subset $A \subseteq S$ so that $Q = S/A$ is passed to Algorithm 1 as a characteristic function $q : \mathbb{R}^2 \to \{0, 1\}$ such that $q(x) = 1$ if and only if $x \in A$.

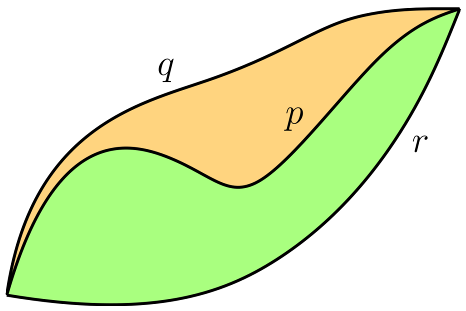

Fig. 4: Example of how given a bounding chain for $p - q$ (orange region) and one for $p - r$ (green region) we obtain a bounding chain for $q - r$ (union of orange and green regions). The fact that the bounding chains have signed area, ensures that when two regions intersect, the area cancels out.

The algorithm proceeds by first letting $qPts$ have $Null$ as its first point, as this will be the image of every point in A and therefore has no (unique) correspondence with a point in S, and then adding every point that is not in A. Secondly in the lines 6–16 it translates the triangles in $tris$ to triangles with the corresponding vertices in $qPts$. This translation is done in lines 9–14, where the indices in a triangle tri are replaced in $qTri$ by their counterparts from $qPts$ unless the point is in A; subsequently if tri contains a point in A the index 0 is added to $qTri$ in line 16, so that it is added at most once. The resulting simplex $qTri$ is then added to $qTris$, and $Q = (qPts, qTris)$ is returned.

Now we proceed to show how we can use the quotient complex and bounding chains to efficiently calculate a distance array using homology area in Algorithm 2. As we note in Section III even though the process of calculating a bounding chain is slow, calculating the distance array can be done efficiently due to Corollary 1.

Algorithm 2 comprises two phases, phase one in lines 1–16 performs the heavy calculations (i.e. it computes the necessary bounding chains). Phase two, in lines 17–27 uses these bounding chains to compute the array of distances between paths.

This algorithm uses the subroutine $BoundingChain(p, q)$ which, given two 1–chains p, q, calculates the bounding chain of $p - q$ in case they are homologous, and reports a failure otherwise. In our implementation, we use a least squares algorithm to solve equation (2) (which we justify in Section II-B), and intuitively, given the innate sparsity of the boundary matrix, an alternative would be to use a simple row-reduction algorithm, but the associated change of base matrix will be dense, making it more computationally expensive.

In phase one the algorithm builds up three lists $PathCluster$, a list that at position i it has the homology class of path $Paths[i]$; $HomologyClusters$, a list of lists where the elements of $HomologyClusters[i]$ are the indices of paths in the i–th homology class. The third list, $BoundingChains$, stores, at position i the results of $BoundingChain(p, q)$ where p is the first path in the i–th homology cluster, and q ranges over the paths homologous to p, in order of appearance in $Paths$.

In phase two, the algorithm uses lists built up in phase one

Algorithm 2: Distance array using homology area.

```
input  : $Paths$ /* a list of paths */
output : An array of pairwise distances
1  $PathCluster = \{\}$;
2  $BoundingChains = \{\}$;
3  $HomologyClusters = \{\}$;
4  for $i \in \{0, \ldots, \#Paths\}$ do
5      for $Cluster \in HomologyClusters$ do
           /* check if the current path is
              in any existing cluster */
6          $p = Paths[i]$;
7          $q = Paths[Cluster[0]]$;
8          $j = Index(Cluster, HomologyClusters)$;
9          $Chain = BoundingChain(q, p)$;
10         if successful then
11             append $j$ to $PathCluster$;
12             append $i$ to $Cluster$;
13             append $Chain$ to $BoundingChains[j]$;
14     if no cluster was found then
           /* not homologous to any already
              seen path, new cluster */
15         append $\{i\}$ to $HomologyClusters$;
16         append $\{\}$ to $BoundingChains$;
   /* use the bounding chains to compute
      pairwise distances */
17 Distarray = $\{\}$;
18 for $i \in \{0, \ldots, \#Paths\}$ do
19     $c = PathCluster[i]$;
20     for $j \in \{i + 1, \ldots, \#Paths\}$ do
21         if PathCluster[j] = $c$ then
22             $k$ = index of $i$ in $HomologyCluster[c]$;
23             $l$ = index of $j$ in $HomologyCluster[c]$;
24             $Chain = BoundingChains[c, k - 1] - BoundingChains[c, l - 1]$;
25             append area($Chain$) to $Distarray$;
26         else
27             append $\infty$ to $Distarray$;
28 return $Distarray$
```

to produce the array of distances $Distarray$. To achieve this it checks what is the homology cluster of path i (line 19) and computes the distance to path j for every $j > i$ by checking if they have the same homology class; if they are, it uses Corollary 1 (which is summarized in Figure 4) to calculate the distance (lines 21–25), otherwise, it stores the value ∞ (line 27).

Now, note that finding the minimum area bounding chain amounts to solving a $k \times l$ linear system, where k, l are respectively the number of edges and faces of the simplicial complex. If we let $\mu = \max(k, l)$, such a system can be solved by a linear solver in $O(\mu^2)$. From analyzing algorithm 2 we see that when analyzing each path, we need to compute at most m bounding chains, where m is the number of elements of $HomologyClusters$ (line 5). Therefore, the complexity of phase 1 is $O(\mu^2 Nm)$ here N is the number of paths. Finally, in each iteration of phase 2, we perform one vector adition, which has complexity $O(\mu)$ and compute the area of the resulting chain. Recall that calculating the area of a bounding chain, amounts to an inner product, and therefore also has complexity $O(\mu)$. Since this is performed for every possible pair of indices, the complexity of phase 2 is $O(N^2\mu)$, which brings the total complexity of Algorithm 2 $O(N^2\mu + \mu^2 NM)$.

In the next section we describe the results that guarantee the correctness of our method.

B. Theoretical framework

Let $p = p_0, p_1, \ldots, p_n$ be an edge path in X, that is, a list of vertices of the X such that $[p_i, p_{i+1}] \in X^{(1)}$. Now we observe, as pointed out in [11], that any such path can be represented by a 1–chain in $C_1(X) = \mathbb{R}^{b_1}$, i.e. a vector of the form $\tilde{p} = (\tilde{p}_0, \ldots, \tilde{p}_{b_1})$ where $\tilde{p}_i$ is the number of times the i–th edge is traversed in the path counted with orientation[4]. Further, note that given two paths $p = p_0, \ldots, p_n, \quad q = q_0, \ldots, q_m$, if $p_0 = q_0$ and $p_n = q_m$ then $\partial(\tilde{p} - \tilde{q}) = 0$.

So, given two edge-paths p, q on a simplicial complex X we say that they are *homologous* if there exists a 2–chain c such that $\partial c = p - q$, which brings us to the following definition:

Definition 1. *Define the homology area path distance on X as*

$$d_{H(X)}(p, q) = \min_{\partial c = p - q} \text{area}(c) \tag{3}$$

where $\min_{x\in\emptyset} x = +\infty$.

Proposition 1. *Let X be a simplicial complex where each 2–simplex has non-negative area, then $d_{H(X)}$ is symmetric and satisfies the triangle inequality, i.e.:*

- $d_{H(X)}(p, q) = d_{H(X)}(q, p)$
- $d_{H(X)}(p, q) + d_{H(X)}(q, r) \geq d_{H(X)}(p, r)$

Proof: Let c be a bounding chain of $p - q$, then due to linearity of ∂, $\partial(-c) = q - p$ and $\text{area}(c) = \text{area}(-c)$ meaning that $d_{H(X)}(p, q) = d_{H(X)}(q, p)$.

Assume p, q and r, q are homologous pairs of paths, then there exist 2–chains $c_{p,q}$ and $c_{r,q}$ such that $\partial c_{p,q} = p - q$ and $\partial c_{q,r} = q - r$. This implies that $\partial(c_{p,q} + c_{q,r}) = p - q + q - r = p - r$ and therefore $c = (c_{p,q} + c_{q,r})$ is a bounding chain for $p - r$. Furthermore we see that $c_i \neq 0$ only if $(c_{p,q})_i \neq 0$ or $(c_{q,r})_i \neq 0$ and therefore $\text{area}(c) \leq \text{area}(c_{p,q}) + \text{area}(c_{q,r})$, which implies $d_{H(X)}(p, r) \leq d_{H(X)}(p, q) + d_{H(X)}(q, r)$. ■

This defines a distance between edge paths that is only finite for paths that are homologous. A shortcoming of this, is that only homologous paths that have the same first and last points may have finite area. However, this can be mitigated using relative homology.

More specifically, we recall from [11], that by defining *goal regions A* in the space X (associated to a subcomplex) and computing the first relative homology group $H_1(X, A)$ we define instead the homology class of 1–chains with boundary contained in A, which correspond to paths beginning and ending in A.

Remark 1. *By Theorem 1 we can compute the homology group $H_1(X/A)$ instead of $H_1(X, A)$ as these are canonically isomorphic. This has a practical advantage, since X/A has less simplices then X, which reduces the dimension of the boundary matrices $C_1(X/A), C_2(X/A)$ relative to those for $C_1(X, A)$ and $C_2(X, A)$, making the computation of solutions to* (3) *more efficient.*

Algorithm 1 shows how one can create a quotient complex from a simplicial complex and the characteristic function of a closed region A.

Now, given a 2–dimensional simplicial complex X and a subcomplex A we note that, any 2–simplex in $\sigma \in (X/A)^2$ is a quotient of some 2–simplex $\sigma' \in X^2$ and hence we can define $\text{Area}(\sigma) = \text{Area}(\sigma')$, which then allows us to implement minimum area homology on X/A.

In [12] we presented a framework to compute a minimum area bounding chain on a general simplicial complex by computing the solution to equation (2), and propose several methods by which this may be achieved. Particularly, computing least squares solutions as this can be done efficiently on sparse systems[5]. However, often fast methods, such as least squares, may yield chains that do not have minimal area. In what follows, we prove that if the chain complex X can be embedded in $\mathbb{R}^2$, then for any 1–cycle there is at most one bounding chain, thereby proving that in such cases approximate methods yield exact solutions. In Proposition 2 we generalize to quotients over sets A with no holes.

Lemma 2. *Any simplicial complex X which can be embedded into $\mathbb{R}^2$ satisfies $H_2(X) = 0$.*

Proof: Let X be a simplicial complex that can be embedded into $\mathbb{R}^2$ and X^c denote (a triangulation of the closure of) its complement, then using the Mayer-Vietoris sequence of $\mathbb{R}^2 = X \cup X^c$, we have:

$$\cdots \to H_3(\mathbb{R}^2) \to H_2(X \cap X^c) \to$$
$$\to H_2(X) \oplus H_2(X^c) \to H_2(\mathbb{R}^2) \to \cdots$$

Now, recall that $H_i(\mathbb{R}^k) = 0 \forall i \neq 0$ ([16]). Note also that $X \cap X^c = \partial X$ is a 1–dimensional subcomplex of X and X^c, so $H_2(X \cap X^c) = 0$ (as it has no 2–cells), hence we conclude that the sequence:

$$0 \to H_2(X) \oplus H_2(X^c) \to 0$$

is exact, and $H_2(X)$ is a summand of 0 and hence $H_2(X) = 0$. ■

Proposition 2. *Let X be a simplicial complex which can be embedded in $\mathbb{R}^2$, and A a subcomplex of X such that $H_1(A) = 0$. Then, given any 1–cycle c, there exists, at most one 2–chain s, such that $\partial s = c$.*

[4]That is, if the i–th edge is of the form $[e, e']$ then $\tilde{p}_i = p_+ - p_-$ where p_+ is the number of times the edge is traversed from e to e', and p_- is the number of times the edge is traversed in the from e' to e.

[5]using for example the `LSQR` algorithm [17].

Proof: To begin, consider the long exact sequence in homology of the pair (A, X):

$$\begin{aligned}\cdots \to H_{i+1}(A) \to H_{i+1}(X) \to \\ \to H_{i+1}(X, A) \to H_i(A) \to \cdots\end{aligned}$$

Particularly for $i = 1$, $H_i(A) = 0$ and so we have:

$$\cdots \to H_2(A) \to H_2(X) \to H_2(X, A) \to 0 \to \cdots$$

So, using Lemma 2 we know that $H_2(A) = H_2(X) = 0$ and so $H_2(X, A) = 0$. Finally, given any 1–cycle c let s, s' be two 2–chains satisfying $\partial s = \partial s' = c$, then $t = s - s'$ is a 2–cycle, and since $H_2(X, A) = 0$, we conclude that t is the boundary of some 3–chain. However, since X can be embedded in $\mathbb{R}^2$, it has no 3–simplices, from which we can conclude that $t = 0$ and $s = s'$. ■

We now show how we can use this result to quickly calculate bounding chains between paths.

Corollary 1. *Let X be a two-dimensional simplicial complex and A a subcomplex with $H_1(A) = 0$, then for three one chains p, q, r such that $c_{p,q}$ and $c_{p,r}$ are the minimum area bounding chains of $p - q$ and $p - r$, respectively, then $(c_{p,r} - c_{p,q})$ is the minimum area bounding chain of $q - r$.*

Proof: Recall from the proof or Proposition 1 that $(c_{p,r} - c_{p,q})$ is a bounding chain of $q - r$, (this comes from linearity of the boundary operator). Furthermore, from Proposition 2 if there is a bounding chain of $q - r$ then it is unique, therefore, $(c_{p,r} - c_{p,q})$ must be the minimum area bounding chain of $q - r$. ■

Finally, since our main goal is to cluster paths on a simplicial complex, we note that many algorithms used for path clustering rely on computing distances between all pairs of paths. Specifically, if we want to cluster the paths $p_0, p_1, \ldots, p_N$ with a distance function d, then we need to compute:

$$\begin{aligned}Distarray = [\, d(p_0, p_1), \ldots, d(p_0, p_N), d(p_1, p_2), \ldots \\ \ldots, d(p_1, p_N), \ldots, d(p_{N-1}, p_N)\,]\end{aligned}$$

Which corresponds to the entries of the matrix $[d(p_i, p_j)]_{i,j=0}^{N}$ that lie above the main diagonal. In the case that the distance function is hard to compute, as it often is in the case of path distances ([18]), producing such a distance array can take a long time. Corollary 1 provides a way to mitigate this cost when using the homology area distance function, and we exploit this fact in Algorithm 2 to avoid calculating unnecessary bounding chains.

III. Experiments

To test our framework we used two distinct environments. First we created a synthetic dataset described in Section III-A and tested our framework on it. Secondly, in Section III-B we used our framework on real data, namely on the set of tracks of pedestrians on the computer science forum at Edinburgh University [19], a subset of which can be seen in Figure 6a.

A. Toy example

The space created for the synthetic dataset comprises a bounded region of the plane with two holes. The simplicial complex was obtained by random sampling on this area. The relative region we chose comprises two rectangles along the side so that we can observe different clusters of paths going left to right.

In order to be able to observe the characteristics of clustering by relative homology previously identified in [11], as well as the distinction between different sub-clusters (using minimum area homology), we synthesized paths satisfying:

- Each path begins in one of two small areas in the relative region to the left (we call them O_1 and O_2).
- Each path ends in one of two small areas in the relative region to the right (we call them T_2 and T_2).
- Each path passes through a preselected (randomly chosen) point in a neighborhood of the middle vertical axis of the space (comprising three connected components which we call M_1, M_2, and M_3).

Paths were then computed by randomly drawing a triple from $O_i \times M_j \times T_k$, and interpolating these points with a Gaussian process. By choosing the paths in this way we have access to a ground truth model which is simply given by the triple (i, j, k).

As expected, we observed that relative homology alone was enough to detect the different clusters depending on j, furthermore, using homology area we were able to fully recover the ground truth, as seen in Figure 5. Homology area allows us to distinguish between the four clusters with $j = 2$ which cannot be separated with homology alone. Furthermore, for comparison we used FastDTW [13] from which we obtained similar results.

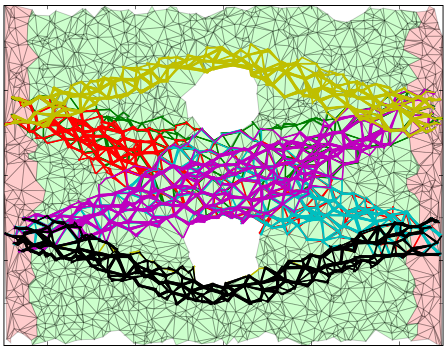

Fig. 5: Randomly generated simplicial complex with two holes (in white), and a set of randomly generated paths connecting the relative regions of the complex.

B. Edinburgh dataset

In this experiment we randomly selected 2000 paths from the dataset of trajectories collected in the computer science forum of Edinburgh University (as seen in Figure 6a). From these paths we constructed a simplicial complex that fully covered the area around them as shown in Figure 6b, and the relative regions were chosen so that they contain the endpoints of most paths (those that were not encompassed

Algorithm	Average time (ms)	Total time (h:m)
Homology Area	90.809 ms	0 h 17 m
FastDTW	5.301 ms	2 h 42 m
DTW	18.152 ms	10 h 21 m

TABLE I: Time elapsed in computing $Distarray$ with DTW, fastDTW, and Homology area. "Average time" denotes the average amount of time to compute the distance between two paths, whereas "Total time" denotes the total amount of time spent in computing the distance array.

were discarded). In Figure 6b we also show the result of clustering the paths of a single homology cluster by homology area where we can observe 4 pronounced clusters (green,red,purple, and light-blue) and 4 outliers (dark blue).

In order to test the performance of our algorithm we computed the array of distances using DTW, FastDTW [13] and our implementation of Homology area, and noted the times in Table I. These tests were performed using a single core of an Intel Core i7–4790K @ 4.00GHz with 32Gb RAM. Our algorithms were fully implemented in Python, and for comparison we used the implementations of DTW and FastDTW from [20]. From these results we observe that Homology area outperforms (in Total time) DTW and FastDTW. The reason for this is that even though it takes more time to calculate each individual pairwise distances with Homology Area, by using Algorithm 2 most of the bounding chains needed to compute the distance array are obtained as linear combination of other bounding chains.

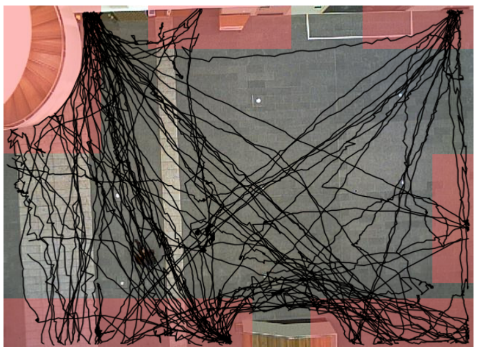

Fig. 6 a: Picture of the Edinburgh computer science forum overlayed with a set of paths from the dataset. The areas shaded in red were used to generate the relative region for the homology area clustering.

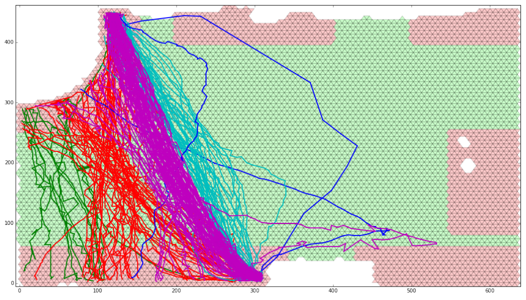

Fig. 6 b: One of the homology clusters of paths from [19] clustered according to homology area. The dark blue paths are each in clusters of their own, the remaining paths fall mainly into 4 clusters.

IV. Conclusions

In this paper we were able to show that we can extend topological methods for path clustering to provide a compatible path distance. Furthermore, we were able to observe that the proposed method is able to cope with large numbers of paths by leveraging the fact that not every distance needs to be computed from scratch, thereby reducing the time needed to produce a full distance array (essential for distance-based clustering) by an order of magnitude.

Acknowledgements: The authors would like to thank Erin Wolf Chambers for her valuable contribution to the research and structure of this paper.

References

[1] OpenStreetMap. [Online]. Available: https://www.openstreetmap.org

[2] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot Programming by Demonstration," in *Springer Handbook of Robotics*. Springer Berlin Heidelberg, 2008.

[3] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena, "From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017.

[4] C. M. Bishop, *Pattern recognition and machine learning*. New York: Springer, 2006.

[5] D. Müllner, "Modern hierarchical, agglomerative clustering algorithms," *arXiv:1109.2378 [cs, stat]*, Sep. 2011.

[6] L. Brun, A. Saggese, and M. Vento, "Dynamic Scene Understanding for Behavior Analysis Based on String Kernels," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 10, Oct. 2014.

[7] S. Atev, G. Miller, and N. Papanikolopoulos, "Clustering of Vehicle Trajectories," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 3, Sep. 2010.

[8] D. Makris and T. Ellis, "Learning semantic scene models from observing activity in visual surveillance," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 35, no. 3, Jun. 2005.

[9] B. Morris and M. Trivedi, "A Survey of Vision-Based Trajectory Learning and Analysis for Surveillance," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 8, Aug. 2008.

[10] E. J. Keogh and M. J. Pazzani, "Scaling Up Dynamic Time Warping for Datamining Applications," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '00. New York, NY, USA: ACM, 2000.

[11] F. T. Pokorny, K. Goldberg, and D. Kragic, "Topological trajectory clustering with relative persistent homology," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016.

[12] E. W. Chambers and M. Vejdemo-Johansson, "Computing minimum area homologies," in *Computer Graphics Forum*, vol. 34. Wiley Online Library, 2015.

[13] S. Salvador and P. Chan, "Toward Accurate Dynamic Time Warping in Linear Time and Space," *Intell. Data Anal.*, vol. 11, Oct. 2007.

[14] S. Atev, O. Masoud, and N. Papanikolopoulos, "Learning Traffic Patterns at Intersections by Spectral Clustering of Motion Trajectories," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2006.

[15] S. Aghabozorgi, A. Seyed Shirkhorshidi, and T. Ying Wah, "Time-series clustering – A decade review," *Information Systems*, vol. 53, oct 2015.

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

[17] C. C. Paige and M. A. Saunders, "LSQR: An Algorithm for Sparse Linear Equations and Sparse Least Squares," *ACM Trans. Math. Softw.*, vol. 8, no. 1, Mar. 1982.

[18] B. Morris and M. Trivedi, "Learning trajectory patterns by clustering: Experimental studies and comparative evaluation," in *IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009*, Jun. 2009.

[19] R. Fischer. [Online]. Available: http://homepages.inf.ed.ac.uk/rbf/FORUMTRACKING/

[20] K. Tanida, "fastdtw." [Online]. Available: https://pypi.python.org/pypi/fastdtw