PERSISTENT HOMOLOGY TRANSFORM OF EMBEDDED KNOTS

ZACHARY OJAKLI* AND AMY SOMERS†

Abstract. We explore the applicability of the Persistent Homology Transform (PHT) to distinguish embedded knots in 3-space. Researchers have shown the PHT is a sufficient statistic to fully recover simple closed curves in two and three dimensions. We test its ability to classify different knots with the application of with different Reidemeister moves. Our results exhibit practical limitations in the ability of the PHT to recover topological data of embedded knots.

1. Introduction

Topological data analysis (TDA) provides us with a set of methods to study the shape of complex data. Persistent homology (PH) is an important tool from TDA which considers the way that the homology groups change as you pass through a filtration that approximates a data set. It records the birth and death times of connected components and holes — represented by homology classes — in the filtration. The persistence of homology classes is recorded in a persistence diagram (PD).

In [12], the authors use the persistent homology transform (PHT) to model simplicial complexes in $\mathbb{R}^d$ as a collection of persistence diagrams parameterized by the sphere S^{d-1}. For each vector $v \in S^{d-1}$, the PHT assigns to it a PD from the directional height filtration in the direction of v. In particular, Turner et al. [12] compute the 0-dimensional PHT of simple closed curves in $\mathbb{R}^2$ from k directions to test the theoretical result that the PHT taken from infinitely many directions can fully recover topological objects. We apply a similar approach to analyze simple closed curves embedded in $\mathbb{R}^3$. In particular, we study knots and investigate if the PHT can identify different embeddings of the same knot while distinguishing different knots. We investigate how factors such as the number of directions and methods of embedding the knot impact the efficacy of this approach.

1.1. Related Work.

Tools from topological data analysis have been used to classify and understand complex shapes in $\mathbb{R}^3$. We are interested in exploring the use of methods from TDA for knot identification. The work of [6] explores how Mapper can be used to analyze point clouds from knots in $\mathbb{R}^3$. Further, principal component analysis (PCA) has been used to analyze the Jones polynomial, a knot invariants [7].

The PHT is an important tool in computational topology. Given a point cloud $K \subset \mathbb{R}^d$, we fix a number of directions $v \in S^{d-1}$, and for each v we associate a PD obtained by scanning K in the direction of v. The authors of [5] show that this transform is injective on the space of shapes and is continuous from the sphere to the space of PDs. PHT is applied in [12] to model shapes in $\mathbb{R}^2$ and surfaces in $\mathbb{R}^3$. In this paper, we specifically apply PHT to classify knots in $\mathbb{R}^3$. In [3] the authors present an algorithm for reconstructing a graph embedded in $\mathbb{R}^d$ with n vertices that uses the persistent homology transform (PHT) from $n^2-n+d+1$ directional augmented persistence diagrams (APDs). An APD is defined to be a PD with the additional data of points that have zero persistence, that is, where birth time equals death time. Points with zero persistence are not included in a standard PD so an APD captures this additional data. This work provides a theoretical guarantee on the number of directional APDs necessary for us to fully recover a knot.

1.2. Research Question.

When defining the Persistent Homology Transform, [12] run the PHT on sets of simple, closed, planar objects embedded in $\mathbb{R}^3$ and classify them.

Definition 1.1. A *knot* K is a smooth embedding of S^1 into $\mathbb{R}^3$.

* *Dartmouth College*
† *University of California, Santa Barbara*
This work is supported by NSF Research and Training Grant DMS-2136090.

Since knots are also simple closed curves sitting in $\mathbb{R}^3$, we want to know if we can take a dataset of several embeddings of unique knots obtained by performing Reidemeister moves and classify them by using the PHT.

2. Background

2.1. **Simplicial Homology.** We will first review simplicial homology. An *n-simplex* is the generalization of the notion of a point, line, triangle, or tetrahedron to n-dimensions. Specifically the convex hull of a collection of $n+1$ affinely independent points $v_0, v_1, \ldots, v_n$ is an *n-simplex* denoted $[v_0, v_1, \ldots, v_n]$. The n-simplex $[v_0, v_1, \ldots, v_n]$ has an orientation and any odd permutation of the vertices negates the orientation, e.g. $[v_n, v_{n-1}, \ldots, v_0] = -[v_0, v_1, \ldots, v_n]$. A *face* of $[v_0, v_1, \ldots, v_n]$ is any simplex $[u_0, u_1, \ldots, u_m]$ with $\{u_0, u_1, \ldots, u_m\} \subset \{v_0, v_1, \ldots, v_n\}$. Each vertex v_i is an example of a 0-simplex and the line segment connecting v_i, v_j for $i \neq j$ is the 1-simplex $[v_i, v_j]$. A *simplicial complex* $\mathcal{K}$ is a collection of simplices such that

(i) every face of a simplex in $\mathcal{K}$ is also in $\mathcal{K}$,
(ii) if $\sigma, \tau \in \mathcal{K}$ have non-empty intersection, then $\sigma \cap \tau$ is a face of both σ and τ.

The boundary of an n-simplex $\sigma = [v_0, v_1, \ldots, v_n]$ is the collection of simplices of dimension $n-1$ on the vertices $v_1, v_1, \ldots, v_n$. The *boundary operator* ∂_n on σ is given by

$$\partial_n(\sigma) := \sum_{i=0}^{n} (-1)^i [v_0, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n].$$

A *n-chain* is a formal linear combination

$$\sum_{i=1}^{k} \alpha_i \sigma_i$$

of n-simplices $\sigma_1, \ldots, \sigma_k$ in a finite simplicial complex $\mathcal{K}$ with coefficients $\alpha_i \in \mathbb{Z}/2\mathbb{Z}$. The collection of all n-chains forms a vector space $C_n(\mathcal{K})$ over $\mathbb{Z}/2\mathbb{Z}$ with basis consisting of all n-simplices $\sigma_1, \ldots, \sigma_k$. Extending the boundary operator linearly defines the *n-th simplicial boundary map*

$$\partial_n : C_n(\mathcal{K}) \to C_{n-1}(\mathcal{K}).$$

We note that the elements of $\operatorname{im} \partial_{n+1}$ are the boundaries of n-simplices in $\mathcal{K}$. The elements of $\ker \partial_n$ are known as *n-cycles*. Since $\partial_{n+1} \circ \partial_n = 0$ for all n, $C(\mathcal{K}) = \bigoplus_n C_n(\mathcal{K})$ is a chain complex.

The *n-th homology group* of $\mathcal{K}$ is

$$H_n(\mathcal{K}) = \ker \partial_n / \mathrm{im} \partial_{n+1}$$

as a quotient of vector spaces.

2.2. **Persistent Homology.** One important tool in TDA is *persistent homology (PH)* which measures changes in homology of a parameterized simplicial complex. Given a point cloud $X = \{x_0, \ldots, x_n\}$ in a metric space, we construct a filtered simplicial complex to allow us to approximate the shape of X.

Definition 2.1. A *filtration* of a simplicial complex $\mathcal{K}$ is a nested sequence simplicial complexes

$$\mathcal{K}_{t_0} \subset \mathcal{K}_{t_1} \subset \mathcal{K}_{t_2} \subset \cdots$$

where $t_0 < t_1 < t_2 < \cdots$ and $\mathcal{K} = \bigcup_t \mathcal{K}_t$.

Two common constructions of filtrations are the Čech filtration and the Vietoris-Rips filtration. In this work, we use the directional height filtration.

Definition 2.2. Let $K \subset \mathbb{R}^d$ be a simplicial complex and let $v \in S^{d-1}$. We define the *directional height filtration* of K in the direction v, denoted $K(v)$, so that for each height parameter t, $K(v)_t$ is the simplicial complex consisting of all simplices in K below the hyperplane orthogonal to v through tv, that is,

$$K(v)_t = \{\sigma \in K : x \cdot v \leq t \text{ for all } x \in \sigma\}.$$

Since the sets $K(v)_t$ and $\{x \in K : x \cdot v \leq t\}$ are homotopy equivalent, they have the same homology groups.

For each simplicial complex $\mathcal{K}_{t_i}$ in a filtration, we can compute its homology. As the parameter t_i grows, holes in the filtration form and get filled in. These holes in the filtration for a range of t_i are represented by the *homology class*. For example a 0-dimensional homology class represents a connected components and a 1-dimensional homology class represents the hole bounded by a loop. A homology class that exists from time s and becomes trivial at time t with $s < t$ is said to have *birth time* s and its *death value* is t. For each $s < t$ the inclusion map $K_s \hookrightarrow K_t$ induces an inclusion in the homology groups which induces a homomorphism

$$g_k^{s,t} : H_k(K_s) \to H_k(K_t).$$

We define the *kth persistence homology group* by

$$H_k(s,t) = \operatorname{im} g_k^{s,t}$$

and this group consists of the homology classes in $H_k(K_s)$ which persist to $H_k(K_t)$.

The data of birth and death of homology classes in a filtration is the *persistent homology (PH)* of the filtration and is summarized in a *persistence diagram (PD)* or *barcode*.

Definition 2.3. Let K be a filtered simplicial complex. The *kth persistence diagram (PD)* of K, denoted $D_k(K)$ is the multi-set of points (b,d) in the extended plane $\overline{\mathbb{R}}^2 = \mathbb{R}^2 \cup \{\infty\}$ with $b < d$ each of which has multiplicity $\dim H_k(a,b)$ and the points on the diagonal with infinite multiplicity.

The *kth augmented persistence diagram (APD)* of K, denoted $\overline{D}_k(K)$, is a PD that additionally includes birth-death pairs (b,d) with $b = d$. The APD includes the additional data of points with zero persistence.

2.3. **Persistent Homology Transform.** Let $\mathcal{D}$ denote the space of persistence diagrams.

Definition 2.4. The *persistent homology transform* of $K \subset \mathbb{R}^d$ is the map

$$\begin{aligned} \mathrm{PHT}(K) : S^{d-1} &\to \mathcal{D}^d \\ v &\mapsto (D_0(K(v)), D_1(K(v)), \ldots, D_{d-1}(K(v))). \end{aligned}$$

The space $\mathcal{D}$ of PDs has many possible choices of metric. We will take the metric to agree with the conventions of [12]. Let X, Y denote two persistence diagrams. Since each consists of a collection of points and countably infinite number of copies of the diagonal, there exist bijections $\varphi : X \to Y$. We define a metric on $\mathcal{D}$ by

$$\operatorname{dist}(X,Y) = \inf_{\varphi : X \to Y} \sum_{x \in Y} \|x - \varphi(x)\|.$$

3. Data

The data for this project consists of several point clouds of varying densities for each unique knot. The knots studied are the 0_1 (unknot), 3_1 (trefoil), 4_1 (figure 8), 5_1, 5_2, and 6_1 knots. Define

$$Y = \{0_1, 3_1, 4_1, 5_1, 5_2\}$$

to be the set of knots that we work with.

Definition 3.1. A *grid diagram* $\mathbb{G}$ is an n by n grid on the plane, marked with n different X markings and n different O markings such that each row/column has exactly one X marking and one O marking, respectively, and no grid cell contains more than one marking.

The knot associated with a grid diagram $\mathbb{G}$ is obtained by drawing oriented segments from O markings to X markings horizontally, and from X markings to O markings vertically, with vertical segments always passing over horizontal segments. An example for the trefoil knot is shown in Figure 1.

For each knot $K \in Y$, we use the Python program SnapPy to obtain a three-dimensional embedding of the knot, realized as a grid diagram of K in the xy-plane with understrands "pulled" down to a fixed z value, as shown in Figure 2. Then, we discretize this closed curve by taking n evenly spaced points along the line segments which compose the knot K. This gives a point cloud which we will call a *Grid Diagram point cloud*.

The Grid Diagram point clouds have corners, and all parts of the knot other than the understrands in the crossings lie in a two dimensional plane, so we decided to investigate how to obtain more smooth embeddings of our knots in $\mathbb{R}^3$. We used the spring layout algorithm in the NetworkX Python package to produce these embeddings [13]. An example is shown in Figure 2. We refer to these as *Spring Layout point clouds*.

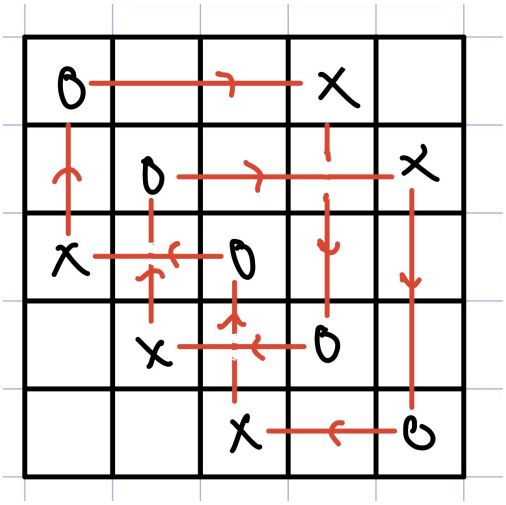

FIGURE 1. Example of a Grid Diagram for the Trefoil

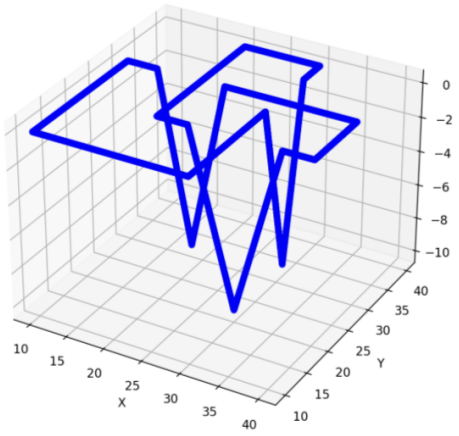

FIGURE 2. Embedding of the Trefoil

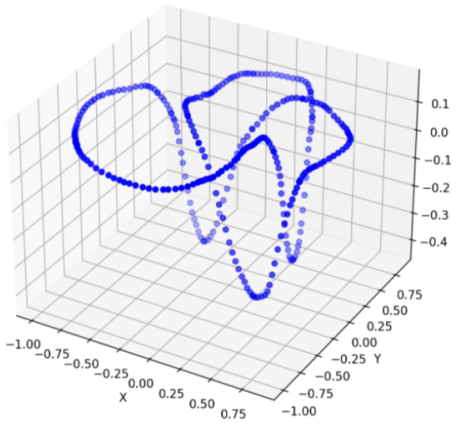

FIGURE 3. Spring Layout Point Cloud for Trefoil ($n = 300$)

For a knot $K \in Y$, we generate an alternate embedding of K by applying m Reidemeister moves of Type I and II using SnapPy. In all of our experiments, we fix $m = 2$ and set the probabilities of performing a Type I or Type II move to be equal. Examples are shown in Figure 3.

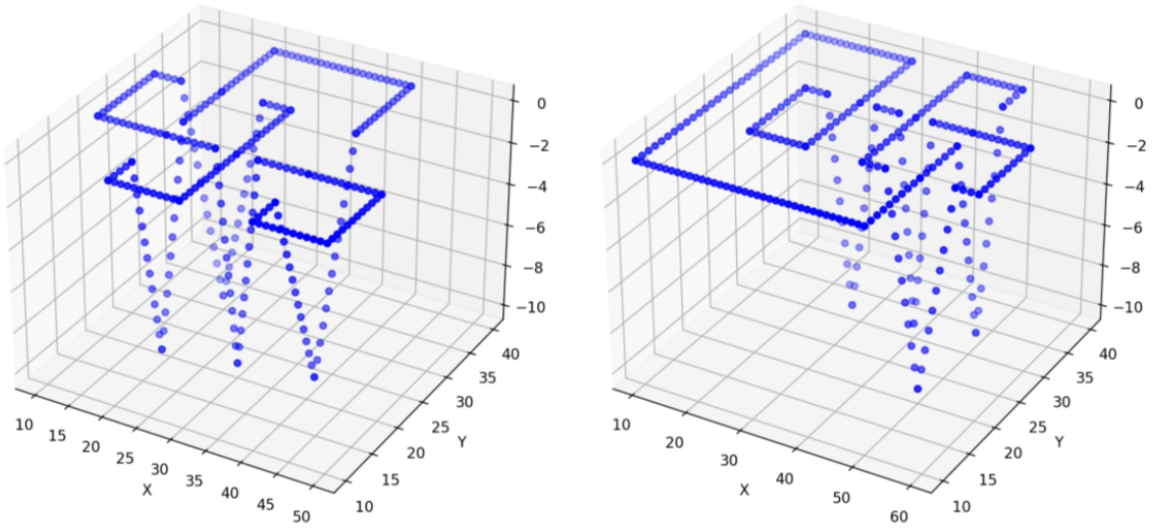

FIGURE 4. Two random Reidemeister moves applied to two Trefoils ($n = 300$)

Now, for each knot $K \in Y$, we perform these 2 random Reidemeister moves to obtain a different embedding K'. We do this 20 times to obtain 21 embeddings of each K. Then, we have a multiset of all knots $\mathcal{Y}$ where $|\mathcal{Y}| = 126$, representing 21 different embeddings of each of the 21 knots we studied. Define a function

$$f : \mathcal{Y} \to \{0, 1, 2, 3, 4, 5\}$$

where $f(K)$ gives the type of the knot $K \in Y$, i.e. the set $\{K \in Y \mid f(K) = 0\}$ is the set of the 21 copies of the unknot.

We will produce three copies of $\mathcal{Y}$, each corresponding to a value of $n \in \{200, 300, 400\}$ where n is the number of points in the point cloud. An example for the trefoil is shown in Figure 4. We will specify the value of n whenever we discuss a specific set $\mathcal{Y}$ or a subset thereof.

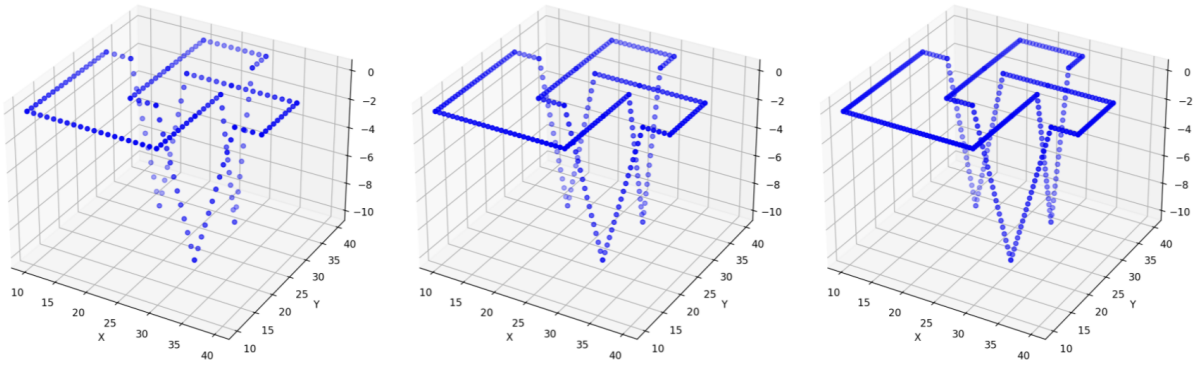

FIGURE 5. Grid Diagram Trefoil Point Clouds for $n = 200$; $n = 300$; $n = 400$

The Sage code for obtaining these point clouds is located in our GitHub repository.

4. Methods

For each of the point clouds generated, we calculate the persistent homology transform from k directions evenly spaced around S^2. Then, we take a union of the k (augmented) persistence diagrams so that we have one PD/APD for each $K \in \mathcal{Y}$. Finally, we cluster these vectors and evaluate the accuracy of the clustering.

4.1. **PHT Calculation.** The code provided in [12] computes the persistent homology transform of simple closed curves in $\mathbb{R}^2$. Since a knot is a simple closed curve embedded in $\mathbb{R}^3$, we made minor modifications to this code. Further, we modified the code to allow us to compute augmented persistence diagrams as well as regular persistence diagrams. We then computed the PHT of each knot from k directions, where in our case, $k \in \{100, 1000, 10000, 18000, 30000\}$. To evenly sample these k directions from the unit sphere, we use the Fibonacci Sphere method. An example is shown in Figure 5

The theoretical guarantee described in [3] tells us that about n^2 directions are needed to perfectly classify each knot for a point cloud with n points. Then, for a point cloud with $n = 200$, approximately 40000 directions should be sufficient in order to perfectly classify each knot. We test fewer directions due to computational constraints.

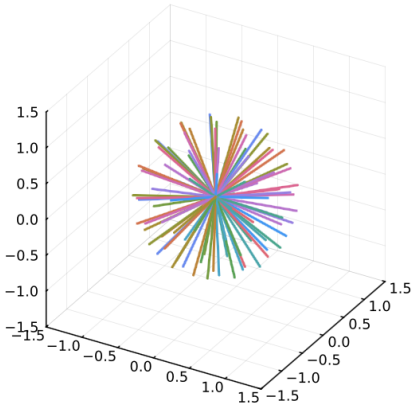

Figure 6. Fibonacci Sphere vectors for $k = 100$

We then take the union of these k persistence diagrams for a particular knot K and put them on a single plot. This gives a single persistence diagram for a particular knot K representing the PHT of K. An example for $k = 10$ is shown in Figure 6.

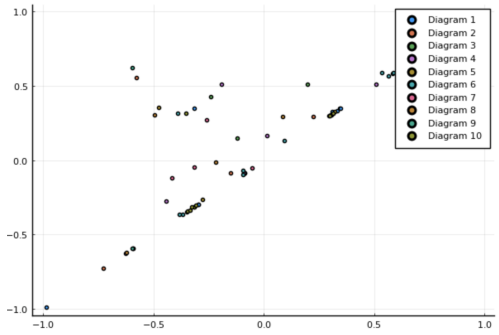

Figure 7. Union of 10 PDs for the Trefoil

4.2. **Featurization & Clustering.** In order to compare sets of persistence diagrams, we need to put them into a form that is amenable to clustering. Two of the most common methods of doing so are using persistence images [1] and persistence landscapes [4]. We focus on using persistence images for computational efficiency.

To produce a persistence image, one takes a weighted sum of Gaussians at each point in the diagram and integrates this over a grid [1]. Persistence images give a stable vector representation of the topological information captured by a persistence diagram. One drawback of obtaining persistence images from a union of augmented persistence diagrams is that we had to modify the weighting function used in [1] in order to keep track of points on the diagonal, so we lose the stability guarantee. An example of a persistence image is shown in Figure 7, where the area of high density in the top left corner corresponds to the large number of points on or near the diagonal [11].

FIGURE 8. Persistence Image for one instance of the Unknot

Define a function

$$\varphi : \mathcal{S} \to \mathbb{R}^d$$

where $\varphi(K)$ is the d-dimensional vector associated with the persistence image of K. Then, the set $V = \{\varphi(K) \in \mathbb{R}^d \mid K \in \mathcal{Y}\}$ represents the 126 vectors corresponding to each knot in the dataset. We then perform k-means clustering on V with 6 clusters. A perfect labelling of the vectors is a labelling in which two knots K_1 and K_2 have the same label if and only if $f(K_1) = f(K_2)$. We compare the labelling given by k-means to this perfect labelling to obtain an accuracy score. This gives a measure of how well the PHT was able to retain the necessary topological data of each knot to classify it correctly.

4.3. **Community Detection.** We also explored an alternate method to grouping the vectors corresponding to each $K \in \mathcal{Y}$, community detection on networks.

This can be done by calculating distances between all vectors pairwise, which we use to create a symmetric similarity matrix. Then, we can view vectors as nodes and add edges between them if the similarity is above some fixed threshold, and analyze the resultant network with Louvain community detection [13].

5. RESULTS

We ran many experiments testing several variables, including:

- Point cloud density ($n \in \{200, 300, 400\}$
- Number of Directions ($k \in \{100, 1000, 10000, 18000, 30000\}$
- Type of Persistence Diagram (Augmented vs. Regular)
- Point Cloud Type (Grid Diagram vs. Spring Layout)

```
Unknot:  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
Trefoil:  [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
Figure 9:  [2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
5_1 Knot:  [3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3]
5_2 Knot:  [4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4]
6_1 Knot:  [5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5]
Accuracy Score:  1.0
```

FIGURE 9. Sample Results for one clustering

As described in 4.2, we assign cluster labels to each $K \in \mathcal{Y}$ and obtain an accuracy score. An sample of results for one clustering with an accuracy of 1.0 is shown in Figure 8.

We also obtain clustering results for a smaller subset $C \subset \mathcal{Y}$ where C contains all the different embeddings of the unknot, trefoil, and 6_1 knot. These knots were chosen because they appear the most different when embedded in $\mathbb{R}^3$. More concretely, they are the three knots with the greatest difference in number of crossings.

We first give results for one particular experiment, where we fix the number of points at $n = 300$, the number of directions at $k = 100$, and use augmented persistence diagrams. We cluster the knots with these criteria twice, once using Grid Diagram point clouds and once using Spring Layout point clouds. The clustering results for this experiment are shown in Figure 9 and Figure 10.

```
[4 4 4 4 3 3 3 2 2 4 2 3 3 4 4 4 3 3 2 4 2]
[3 3 2 0 2 5 2 5 5 3 5 3 3 3 3 0 0 3 3 3 5]
[0 3 3 3 5 0 0 3 2 0 0 2 3 5 0 2 2 0 1 5 0]
[5 5 5 3 5 5 2 5 5 5 1 5 1 5 5 5 5 1 5 1 5]
[3 3 1 3 3 3 2 1 0 0 5 3 3 2 2 0 3 2 0 1 3]
[3 3 3 2 3 5 3 1 5 2 2 1 0 3 3 5 1 3 0 1 1]
Accuracy Score: 0.40476190476190477
```

```
[2 5 5 2 5 2 5 2 5 2 5 2 2 2 5 2 5 5 5 5 2]
[5 3 1 5 3 0 3 0 0 5 5 3 3 0 5 0 5 0 3 0 0]
[0 3 3 0 3 0 0 3 0 1 3 3 3 3 3 0 3 0 0 0 3]
[3 1 3 0 3 3 4 0 1 4 3 3 0 3 0 4 1 3 3 4 0]
[0 3 4 4 3 3 3 3 3 4 0 3 3 3 1 3 1 4 4 3 4]
[3 1 4 1 1 3 0 3 0 1 1 1 1 1 0 4 3 0 1 3 4]
Accuracy Score: 0.3888888888888889
```

FIGURE 10. Clustering and accuracy scores for Grid Diagram point clouds (left) and Spring Layout point clouds (right)

```
[2 2 2 2 2 2 2 2 2 2 0 2 2 2 2 2 2 2 2 2 2]
[2 2 0 2 0 1 0 1 1 2 1 2 2 2 1 2 2 2 2 2 1]
[1 1 1 2 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1]
Accuracy Score: 0.6507936507936508
```

```
[2 1 1 1 1 1 1 2 1 2 1 1 1 1 1 2 1 1 1 1 2]
[1 0 0 1 0 0 0 1 1 1 1 0 0 0 1 0 1 0 0 0 1]
[1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
Accuracy Score: 0.5396825396825397
```

FIGURE 11. Clustering and accuracy scores for Grid Diagram (top); Spring Layout point clouds (bottom) for Unknot, Trefoil, 6_1 knot

We saw higher accuracy scores for Spring Layout point clouds on a number of experiments, so we focused our attention on other variables. The results given in 1 and 2 all use Spring Layout point clouds.

In Experiments (1b), (2b), and (3b), where all variables are fixed with $k = 100$ except for n, we see that $n = 200$ gives the lowest accuracy and $n = 400$ gives the highest accuracy for the 6 knots clustering, and $n = 300$ gives the highest accuracy for the 3 knot clustering. In Experiments (1b), (2b), and (3b), where $k = 1000$ and n varies, we again see that $n = 400$ gives the highest accuracy scores. Similar results hold when looking at APDs and varying n. In general, we see that higher values of n give stronger results.

When holding all variables fixed and varying number of directions k, we generally see that higher accuracy scores correspond with higher values of k as expected. This trend is more pronounced in the clustering of the unknot, trefoil, and 6_1 knot.

When varying the type of persistence diagram, we see that APDs give higher accuracy scores for larger values of k, while PDs in some cases give higher scores for smaller values of k.

Experiment No.	Points (n)	No. Directions	PD/APD	Accuracy Score
1a	200	100	APD	0.3651
1b	200	100	PD	0.3730
1c	200	1000	APD	0.3810
1d	200	1000	PD	0.3730
1e	200	10000	PD	0.3810
2a	300	100	APD	0.4048
2b	300	100	PD	0.4048
2c	300	1000	APD	0.3968
2d	300	1000	PD	0.3730
2f	300	10000	PD	0.3968
3a	400	100	APD	0.4523
3b	400	100	PD	0.4523
3c	400	1000	APD	0.4127
3d	400	1000	PD	0.4127

TABLE 1. Table of clustering results for 6 knots

Experiment No.	Points (n)	No. Directions	PD/APD	Accuracy Score
1a	200	100	APD	0.5397
1b	200	100	PD	0.5873
1c	200	1000	APD	0.5556
1d	200	1000	PD	0.5873
1e	200	10000	PD	0.6032
2a	300	100	APD	0.6508
2b	300	100	PD	0.6508
2c	300	1000	APD	0.6508
2d	300	1000	PD	0.5873
2f	300	10000	PD	0.6508
3a	400	100	APD	0.5714
3b	400	100	PD	0.5714
3c	400	1000	APD	0.5714
3d	400	1000	PD	0.5714

TABLE 2. Table of clustering results for Unknot, Trefoil, 6_1 Knot

The results from these experiments show that, as expected, denser point clouds and taking the PHT from a larger number of directions corresponds with higher accuracy scores. To a lesser degree, it seems that APDs are preferable over PDs in terms of accuracy scores when k is large, although the time complexity of running PHT with APDs is much higher than that for PDs.

5.1. **Community Detection.** We run community detection on two of the best performing experiments in the dataset, namely Experiments (2c) and (3c). We create the network by the method described in 4.3, an example of which is shown in Figure 12

These two networks give the community labels for each of the six knots shown in Figure 13:

We do not fix a number of communities to assign nor a maximum/minimum number of vectors to be assigned to a single community. Thus, many of the vectors fall under one community. However, these results are significant because each of the six knots in both networks have some embeddings which are placed into unique communities. That is, for every knot K, some embeddings of K have community labellings which no other knot has. For example, in the community labellings for Experiment (3c), communities 8 and 10

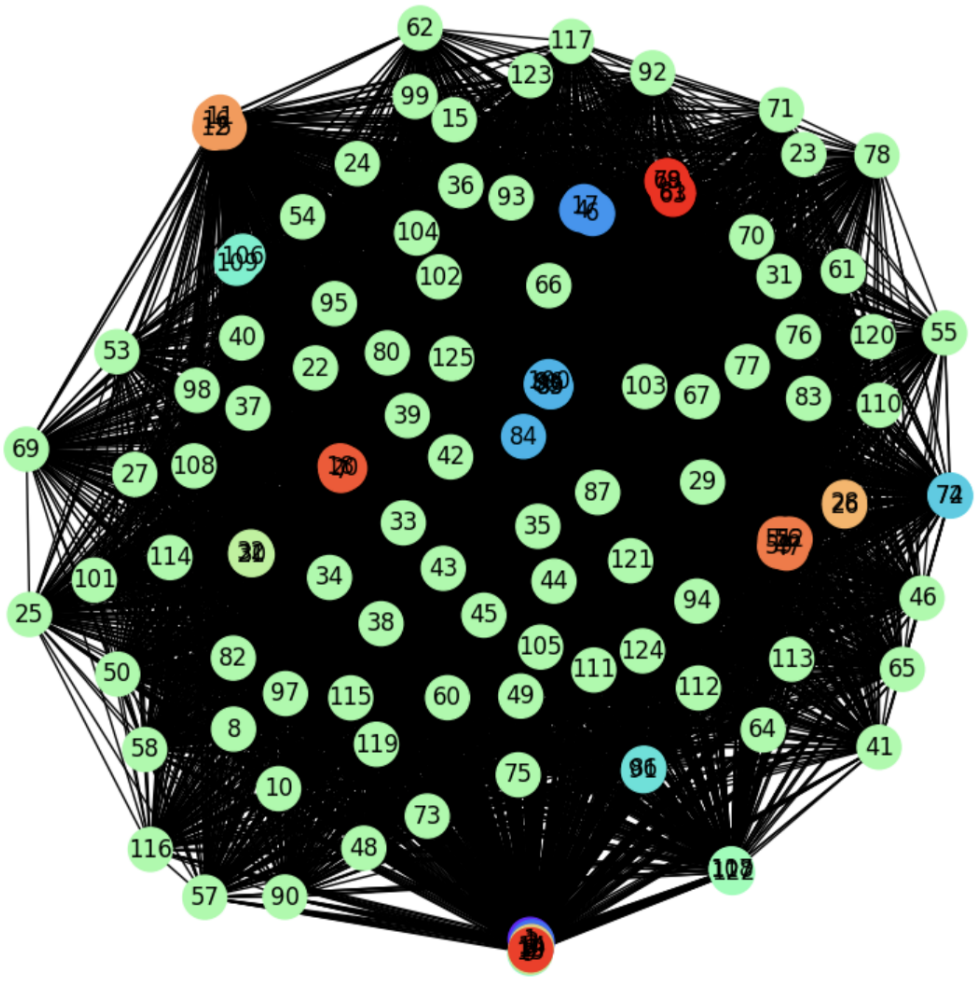

FIGURE 12. Network for vectors from Experiment (3c)

only appear in the labellings for the 6_1 knot (Knot 6), and they appear 2 and 3 times in these labellings, respectively.

6. Discussion

To classify sets of shapes, Turner et al. use an algorithm to orient the shapes in a particular way so as to directly compare directions when taking the PHT [12]. Using Grid Diagram point clouds initially seemed to intuitively mimic this approach by setting a standard by which our simple, closed curves will sit in $\mathbb{R}^3$, making comparison of persistence diagrams from different directions more straightforward.

However, our original intuition for this project was that if one performs a Reidemeister Type I move to a knot K, we obtain an equivalent knot K' with an additional crossing. This knot is isotopically equivalent to one where there exists some space in between the overstrand and understrand of this crossing. If we view K' in the direction of this "space", the knot will look just like K. This intuition follows for Reidemeister moves of Type II and Type III. Thus, we thought there should be some directions of the PHT which give similar persistence diagrams between different embeddings of K. With Grid Diagram point clouds, this idea is somewhat lost because of the rigidity of the grid diagram embedding, prompting us to more smoothly embed the point clouds with Spring Layout. This approach did result in higher accuracy scores, as shown in Figure 9 and Figure 10.

```
Knot 1:
[0, 1, 2, 3, 4, 16, 4, 18, 11, 9, 11, 16, 16, 13, 14, 11, 16, 4, 18, 19, 18]
--------------------------------------------------
Knot 2:
[12, 11, 11, 11, 11, 15, 11, 15, 11, 12, 11, 12, 11, 11, 11, 11, 11, 11, 11, 11]
--------------------------------------------------
Knot 3:
[11, 11, 11, 11, 11, 17, 11, 11, 11, 17, 17, 11, 11, 11, 17, 11, 11, 17, 11, 11, 11]
--------------------------------------------------
Knot 4:
[20, 11, 11, 11, 11, 20, 11, 11, 11, 6, 11, 6, 11, 11, 11, 11, 20, 11, 20, 11, 11]
--------------------------------------------------
Knot 5:
[5, 5, 7, 11, 5, 5, 11, 7, 11, 11, 11, 11, 5, 11, 11, 11, 5, 11, 11, 11, 11]
--------------------------------------------------
Knot 6:
[11, 8, 10, 11, 8, 11, 11, 11, 11, 11, 11, 11, 11, 10, 11, 11, 11, 10, 11, 11, 11]
--------------------------------------------------


Knot 1:
[0, 1, 20, 20, 20, 20, 20, 3, 20, 4, 5, 6, 3, 20, 20, 20, 7, 20, 8, 20, 9]
--------------------------------------------------
Knot 2:
[10, 11, 20, 12, 20, 20, 13, 20, 20, 20, 20, 20, 20, 14, 15, 20, 20, 16, 17, 20, 20]
--------------------------------------------------
Knot 3:
[20, 20, 18, 20, 20, 18, 20, 20, 20, 20, 20, 20, 20, 20, 19, 20, 18, 20, 20, 19, 20]
--------------------------------------------------
Knot 4:
[21, 22, 21, 20, 23, 20, 20, 20, 20, 21, 20, 21, 23, 20, 20, 22, 20, 21, 20, 20, 20]
--------------------------------------------------
Knot 5:
[24, 20, 24, 20, 20, 20, 25, 25, 20, 20, 24, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20]
--------------------------------------------------
Knot 6:
[20, 26, 20, 20, 20, 20, 20, 26, 20, 2, 20, 20, 2, 20, 20, 20, 20, 20, 20, 20, 2]
--------------------------------------------------
```

FIGURE 13. Community labels for vectors from Experiment (3c) (top); Experiment (4c) (bottom)

7. Future Directions

Due to the many computational constraints we encountered in this project, many of the future directions involve running similar experiments with parameters we were not able to test.

First, testing a larger set of knots $\mathcal{Y}$ with more random embeddings for each knot K would give more reliable and replicable results. If each point cloud has n points, it would be most instructive to run the PHT on this larger dataset from closer to n^2 directions, as this directly follows from the theoretical guarantee given by Belton et al. (cite) for closed curves in $\mathbb{R}^2$. Due to computational constraints, we were not able to guide or choice of number of directions with his theoretical guarantee.

Next, other featurization methods could be studied, such as persistence landscapes. The method of matching introduced in 4.3 for community detection could be extended to all APDs of each knot. That is, for two sets of APDs $P_i = \{K_{i,k}, \cdots, K_{i,k}\}$ and $P_j = \{K_{j,k}, \cdots, K_{j,k}\}$ where P_i and P_j correspond to two embeddings K_i and K_j, where k is the number of directions, we can use a metric defined on persistence diagrams (cite) to compare all persistence diagrams pairwise and compute an "optimal bijection" between PHTs. This would allow us to compare specific directions more concretely, and this approach more follows the close attention paid to specific directions, as done in [12].

Another possible direction is to test a larger dataset, which would have a particular benefit in allowing us to implement a supervised machine learning algorithm to classify the knots. If the theoretical guarantees hold and knots are correctly classified by the PHT for a larger dataset, we could try varying the number of Reidemeister moves performed on each K.

The results suggest that taking PHT of an embedded knot is more accurate with denser point clouds, a large amount of directions, using Augmented Persistence Diagrams, and running it on Spring Layout point clouds. We infer that the marginal benefit of increasing the density of the point clouds n will taper off at some value of n, at which point further increases will not be computationally worthwhile. Another future direction could be to find this threshold and quantify it somehow. Also, further experiments could more firmly establish the slight advantage of Augmented Persistence Diagrams over regular PDs that we found.

Acknowledgements

This work was completed as a part of the Geometry and Topology REU program at UCLA, supported by National Science Foundation Research and Training Grant DMS-2136090. We thank Nathan Dunfield for his help in obtaining the embeddings of the knots with SnapPy. We also thank our mentors Mason Porter, Sarah Tymochko, and Sidhanth Raman for all of their guidance and support throughout this project.

References

[1] Henry Adams et al. "Persistence Images: A Stable Vector Representation of Persistent Homology". In: *Journal of Machine Learning Research* 18.8 (2017), pp. 1–35. URL: http://jmlr.org/papers/v18/16-337.html.

[2] Erik Amézquita et al. "Measuring hidden phenotype: quantifying the shape of barley seeds using the Euler characteristic transform". In: *in silico Plants* 4.1 (2021), pp. 1–15. URL: https://doi.org/10.1093/insilicoplants/diab033.

[3] Robin Lynne Belton et al. "Reconstructing embedded graphs from persistence diagrams". In: *Computational Geometry* 90 (2020), p. 101658. ISSN: 0925-7721. DOI: https://doi.org/10.1016/j.comgeo.2020.101658. URL: https://www.sciencedirect.com/science/article/pii/S0925772120300523.

[4] P. Bubenik. "Statistical topological data analysis using persistence landscapes". In: *Journal of Machine Learning Research* 16 (Jan. 2015), pp. 77–102.

[5] Justin Curry, Sayan Mukherjee, and Katharine Turner. *How Many Directions Determine a Shape and other Sufficiency Results for Two Topological Transforms*. 2021. arXiv: 1805.09782 [math.AT]. URL: https://arxiv.org/abs/1805.09782.

[6] Paweł Dłotko, Davide Gurnari, and Radmila Sazdanovic. *Mapper-type algorithms for complex data and relations*. 2023. arXiv: 2109.00831 [math.AT]. URL: https://arxiv.org/abs/2109.00831.

[7] Jesse S F Levitt, Mustafa Hajij, and Radmila Sazdanovic. *Big Data Approaches to Knot Theory: Understanding the Structure of the Jones Polynomial*. 2019. arXiv: 1912.10086 [math.GT]. URL: https://arxiv.org/abs/1912.10086.

[8] Samuel Mickas. "Searching and reconstruction: Algorithms with topological descriptors". In: *Montana State University* (2020). URL: https://www.proquest.com/dissertations-theses/searching-reconstruction-algorithms-with/docview/2404677419/se-2.

[9] Elizabeth Munch. "An Invitation to the Euler Characteristic Transform". In: *arXiv* (2023). URL: https://arxiv.org/pdf/2310.10395.

[10] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[11] Nathaniel Saul and Chris Tralie. *Scikit-TDA: Topological Data Analysis for Python*. 2019. DOI: 10.5281/zenodo.2533369. URL: https://doi.org/10.5281/zenodo.2533369.

[12] Katharine Turner, Sayan Mukherjee, and Doug M Boyer. "Persistent Homology Transform for Modeling Shapes and Surfaces". In: *Information and Inference: A Journal of the IMA* 3.4 (2014), pp. 310–344. URL: https://doi.org/10.1093/imaiai/iau011.

[13] G. Varoquaux, T. Vaught, and Millman. J. "Exploring network structure, dynamics, and function using NetworkX". In: *Proceedings of the 7th Python in Science Conference* (2008), pp. 11–15.