Foundations of Data Science
©American Institute of Mathematical Sciences
Volume **4**, Number **1**, March **2022**
doi:10.3934/fods.2021033
pp. **1–36**

CAPTURING DYNAMICS OF TIME-VARYING DATA VIA TOPOLOGY

Lu Xian

School of Information
University of Michigan
Ann Arbor, MI 48109, USA

Henry Adams

Department of Mathematics
Colorado State University
Fort Collins, CO 80523, USA

Chad M. Topaz

Department of Mathematics and Statistics
Williams College
Williamstown, MA 01267, USA

Lori Ziegelmeier*

Department of Mathematics, Statistics, and Computer Science
Macalester College
Saint Paul, MN 55105, USA

(Communicated by Farzana Nasrin)

Abstract. One approach to understanding complex data is to study its shape through the lens of algebraic topology. While the early development of topological data analysis focused primarily on static data, in recent years, theoretical and applied studies have turned to data that varies in time. A time-varying collection of metric spaces as formed, for example, by a moving school of fish or flock of birds, can contain a vast amount of information. There is often a need to simplify or summarize the dynamic behavior. We provide an introduction to topological summaries of time-varying metric spaces including vineyards [19], crocker plots [55], and multiparameter rank functions [37]. We then introduce a new tool to summarize time-varying metric spaces: a *crocker stack*. Crocker stacks are convenient for visualization, amenable to machine learning, and satisfy a desirable continuity property which we prove. We demonstrate the utility of crocker stacks for a parameter identification task involving an influential model of biological aggregations [57]. Altogether, we aim to bring the broader applied mathematics community up-to-date on topological summaries of time-varying metric spaces.

2020 *Mathematics Subject Classification.* 37N99, 55N31, 62R40, 92B99.

Key words and phrases. Topological data analysis, computational persistent homology, dynamics, mathematical models, machine learning.

L.X. was funded by Macalester College through a grant to L.Z. H.A. was supported by NSF grant 1934725, DELTA: Descriptors of Energy Landscapes by Topological Analysis. C.M.T. was supported by NSF grant DMS-1813752, Variational and Topological Approaches to Complex Systems. L.Z. was supported by NSF grant CDS&E-MSS-1854703, Exact Homological Algebra for Computational Topology (ExHACT).

* Corresponding author: Lori Ziegelmeier.

1. **Introduction.** Drawing from subfields within mathematics, applied mathematics, statistics, and computer science, topological data analysis (TDA) is a set of approaches that helps one understand complex data by studying its shape. The application of TDA has contributed to the understanding of problems and systems in the natural sciences, social sciences, and humanities, including granular materials [32], cancer biology [21], development economics [5], political science [29], urban analytics [30], natural language processing [59], and much more. Classic works that build and review the fundamental ideas of TDA include [60, 34, 31, 26, 9, 48]. While TDA was originally developed with the study of static data in mind, in recent years, it has found fruitful application to time-evolving data, or as we will say, *dynamically-varying* or *time-varying metric spaces*. For example, for a biological aggregation such as an insect swarm, the metric space of interest might be the positions and velocities of all organisms, which vary from frame to frame in the movie of an experimental trial [56]. For networked oscillators, the metric space of interest might be the phase of each oscillator in the ensemble, which, similarly, evolves in time [54]. These systems can produce massive amounts of data, and so there is sometimes a need to simplify or summarize the dynamic behavior of time-varying systems. Here, TDA plays a role.

In this paper, we have three overarching goals. First, we aim to provide a lay reader in the data science community with an overview of topological tools for studying time-varying metric spaces. Second, we demonstrate an application of some of these tools to a parameter recovery problem arising in the study of collective behavior. Finally, we present a new tool for time-varying metric spaces — a crocker stack — and explore its continuity properties. Overall, we hope that our work will provide a mechanism to bring newcomers to the field up-to-date on approaches to time-varying metric spaces, including our own new contribution.

Topological methods for studying time-varying data are built on a technique called persistent homology. *Homology* provides a way to (partially) characterize the topology of an object. The characterization comes in the form of quantities called *Betti numbers* β_k, where the k indicates a boundary of dimension k enclosing a void of dimension $k+1$. Concretely, the number of connected components is β_0, the number of topological loops (circles) is β_1, the number of trapped volumes is β_2, and so on up in dimension. For example, suppose we have a filled-in disk next to a hollow square next to a two-torus, that is, a hollow donut; see Figure 1. The filled-in disk has Betti numbers $(\beta_0, \beta_1, \beta_2, \beta_3, \ldots) = (1, 0, 0, 0, \ldots)$ because it is one object that is contractible and has no higher dimensional structure. The hollow square has Betti numbers $(1, 1, 0, 0, \ldots)$ because it is one object enclosing a flat void. And finally, the torus has Betti numbers $(1, 2, 1, 0, \ldots)$ because it is one object, is generated by two independent circles (one passing around the equator of the donut and one passing around the donut's hole), and encloses an empty volume. Altogether, then, we have $(\beta_0, \beta_1, \beta_2, \beta_3, \ldots) = (3, 3, 1, 0, \ldots)$ if we consider the homology of the union of these three shapes.

In the example above, we have used idealized shapes such as a disk, hollow square, and torus. However, we want to also think about the topological properties of a data set, rather than only idealized shapes. Suppose we have N data points in $\mathbb{R}^m$ and we want to know if this set of points has structure that we cannot see by eye. We can build a *simplicial complex* (in particular, we describe the process to construct a *Vietoris-Rips* simplicial complex, but others exist as well) out of the data by placing an m-dimensional ball of radius $\varepsilon/2$ around each point, forming a k-simplex

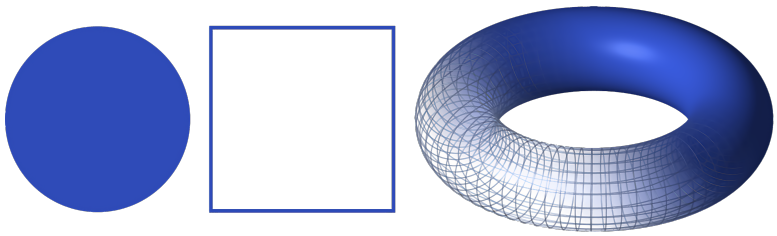


FIGURE 1. A filled-in disk (left) has Betti numbers $(\beta_0, \beta_1, \beta_2, \beta_3, \ldots) = (1, 0, 0, 0, \ldots)$. A hollow square (center) has Betti numbers $(1, 1, 0, 0, \ldots)$. A hollow two-torus (right) has Betti numbers $(1, 2, 1, 0, \ldots)$. If we consider the union of these three shapes, the Betti numbers are $(\beta_0, \beta_1, \beta_2, \beta_3, \ldots) = (3, 3, 1, 0, \ldots)$. Image of the torus taken from Wikimedia Commons https://commons.wikimedia.org/wiki/File:Torus.svg, available for reuse under CCA BY-SA 3.0.

whenever $k+1$ points are pairwise within ε (*i.e.* the balls pairwise intersect), and then, calculating the Betti numbers of the object formed. Of course, the values of the Betti numbers will depend on our choice of ε. For ε approaching zero, all the balls will still be separated, and we have $\beta_0 = N$ with no higher dimensional topological features. For ε approaching infinity, all the balls will overlap and we will have a giant, solid mass with $\beta_0 = 1$ and no higher dimensional features. At intermediate values of ε, the structure of the simplicial complex may well be sensitive to ε, and one may see topological holes of various dimensions that are born and die as ε varies. The *persistent* part of persistent homology refers to calculating homology over a range of values of ε and studying how topological features persist or vary.

Thus far, we have been discussing static data. Three topological summaries of time-varying data are vineyards [19], crocker plots [55], and multiparameter rank functions [37]. As we will explain in more detail later, a *vineyard* is a 3D representation of persistent homology over time. A *crocker plot* is a 2D image that displays the topological information at all scales and times simultaneously, albeit in a manner that does not necessarily elucidate persistence. Finally, a multiparameter rank function is a computable invariant with appealing theoretical properties, providing lower bounds on strong notions of distance between dynamic metric spaces. The three topological summaries we have mentioned here will be presented in more detail in Section 3.

We introduce a new topological summary called a *crocker stack*, a 3D data structure that is an extension of the 2D crocker plot. The crocker stack has three important and useful features. First, as we will prove in Section 7, it satisfies a continuity property. Roughly, a small perturbation of time-varying data results in a small perturbation of the crocker stack. Second, the crocker stack inherits appealing interpretability properties of the crocker plot. A crocker stack is convenient for visualization purposes since topological information for all times and at all scales in the data set is displayed as a single 2D plot, and a third dimension captures the persistence of topological features; see Figure 2 for an example. Third, a crocker stack can be discretized and converted into a feature vector, which can be used as input to machine learning tasks. We explain crocker stacks in detail in Section 4.

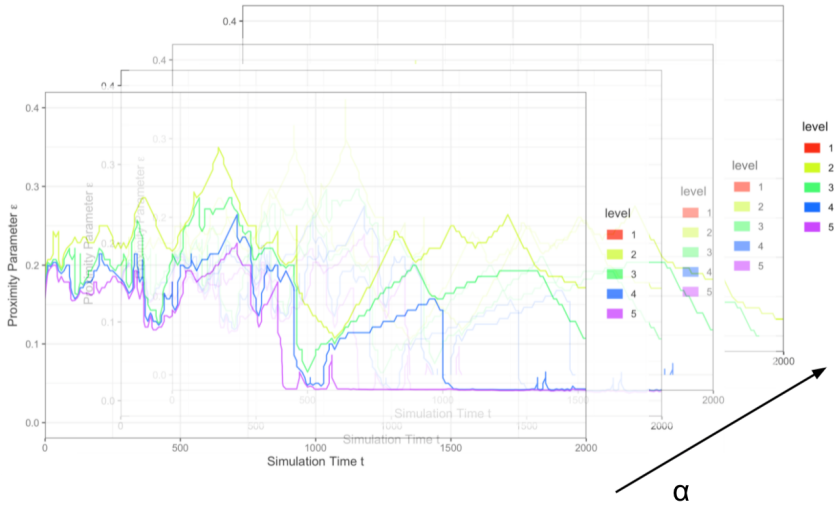


FIGURE 2. An illustrative example of a crocker stack for H_0 computed from a simulation from the Viscek model with noise parameter $\eta = 0.02$; see Section 5. The variable α is a smoothing parameter which captures the persistence of topological features.

The rest of this paper is organized as follows. Section 2 synthesizes information on persistence modules and metric spaces, thus providing the basic topological background underpinning this work in TDA. Section 3 reviews existing frameworks for studying time-varying metric spaces, namely vineyards, crocker plots, and multiparameter rank functions. Section 4 presents our new topological summary, crocker stacks. In Section 5, we use crocker plots and crocker stacks to study a seminal model of collective behavior: the Vicsek model [57]. Specifically, we show that when used as inputs to machine learning algorithms for a parameter recovery task, crocker plots and stacks outperform more traditional summaries of dynamical behavior, drawn from physics. Section 6 reviews notions of distances between metric spaces and persistence modules, which are necessary background for Section 7, in which we study the continuity properties of crocker stacks. We conclude and describe possible future work in Section 8.

2. **Preliminaries.**

2.1. **Persistence modules.**

The construction of persistent homology (as detailed in [9, 26, 31]) begins with a *filtration* $X(0) \subseteq X(1) \subseteq \ldots \subseteq X(n)$, a nested sequence of topological spaces. These spaces are often simplicial complexes; we identify simplicial complexes with their geometric realizations. As an example, a sequence of *Vietoris–Rips complexes* parameterized by scale parameter ε_i is a filtration. Given a fixed point cloud X (or, more generally, a metric space X) and an increasing sequence of parameter values ε_i for $i \in \{0, 1, \ldots, n\}$, denote a sequence of Vietoris–Rips complexes as $X(i) := \mathrm{VR}(X; \varepsilon_i)$ as i varies. Here, the Vietoris–Rips complex $\mathrm{VR}(X; \varepsilon_i)$ at scale ε_i is the abstract simplicial complex with vertex set X, and with k-simplices corresponding to $k+1$ points in X which are pairwise within distance

ε_i. For $\varepsilon_0 \leq \varepsilon_1 \leq \ldots \leq \varepsilon_n$, we have inclusions

$$\mathrm{VR}(X;\varepsilon_0) \hookrightarrow \mathrm{VR}(X;\varepsilon_1) \hookrightarrow \ldots \hookrightarrow \mathrm{VR}(X;\varepsilon_n).$$

We apply k-dimensional homology H_k (with coefficients in a field) to such a filtration. This generates a vector space $H_k(X(i))$ for each space $X(i)$, whose dimension (or rank) is the k-th Betti number β_k. Furthermore, the application of homology assigns, to each inclusion between topological spaces, a linear map between homology vector spaces [60].

The k-dimensional persistent homology of the filtration $X(0) \subseteq X(1) \subseteq \ldots \subseteq X(n)$ refers to the image of induced homomorphisms $H_k(X(i)) \to H_k(X(j))$ for any $i \leq j$. The sequence $H_k(X(1)) \to H_k(X(2)) \to \ldots \to H_k(X(n))$ is a *persistence module* denoted by V, with $V(i) = H_k(X(i))$ for all i. The persistence module for a fixed homological dimension k is known as the *k-dimensional persistent homology* (PH) of a filtration. Persistent homology crucially relies on the fact that homology is a *functor*, which means that an inclusion $X(i) \hookrightarrow X(j)$ indeed induces a map $H_k(X(i)) \to H_k(X(j))$ on homology. More generally, we refer to any sequence of vector spaces and linear maps $V(0) \to V(1) \to \ldots \to V(n)$ as a *persistence module* V, whether or not the vector spaces arise from homology.

2.2. **Persistence diagrams and barcodes.** Persistence diagrams and barcodes each provide a way to display the evolution of topological features in a filtration. In the former, a collection of points in the extended plane $\mathbb{R}^2$ is drawn. If a homology class is born at $X(i)$ and dies at $X(j)$, we represent this homology class by a single point at the two coordinates (i, j) in the *persistence diagram*; see Figure 3 (Left). Since we have $i \leq j$ for all such points (i, j), these points lie on or above the diagonal. Points which remain throughout the entirety of the filtration are said to last to infinity. (A technical point is that for later convenience when defining bottleneck distances in Definition 2.1, the persistence diagram also includes each point along the diagonal, which can be interpreted as a feature that is born and dies simultaneously, with infinite multiplicity.) We denote a persistence diagram of a persistence module V as $\mathrm{Dgm}_k(V)$ for each homology dimension k.

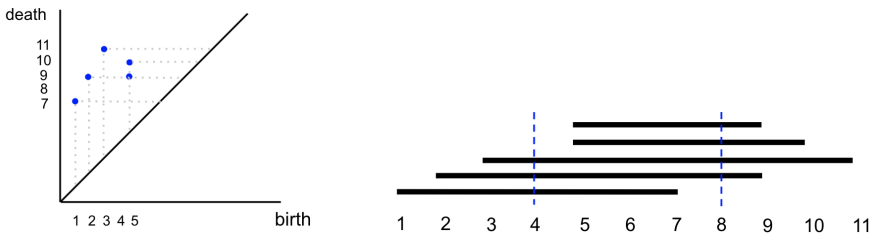


FIGURE 3. Persistence diagram (Left) and corresponding persistence barcode (Right). Let the persistence barcode V consist of the intervals $[1, 7]$, $[2, 9]$, $[3, 11]$, $[5, 10]$, $[5, 9]$, and let $4 = i \leq j = 8$. Then $\mathrm{rank}(V(4) \to V(8))$ is two since there are two intervals in the persistence diagram that contain the interval $[i, j] = [4, 8]$.

Similarly, in the *barcode* representation, there is a distinct *interval* (*i.e.* a bar) corresponding to a homology class persisting over a range of scales; see Figure 3 (Right). The interval begins at scale i, when the feature is born, and ends at scale j, when

the feature dies. The Betti number β_k at scale i is the number of distinct bars that intersect the vertical line through i.

2.3. **The rank invariant.** For $i \leq j$, the *rank* of the map $V(i) \to V(j)$, denoted $\operatorname{rank}(V(i) \to V(j))$, is the number of intervals in the persistence barcode that contain the interval $[i, j]$. In other words, $\operatorname{rank}(V(i) \to V(j))$ is the number of features that are born before scale i and die after scale j. For example, suppose the persistence barcode V consists of the intervals $[1, 7]$, $[2, 9]$, $[3, 11]$, $[5, 10]$, $[5, 9]$, and let $4 = i \leq j = 8$ (see Figure 3). Then $\operatorname{rank}(V(4) \to V(8))$ is two since there are two intervals ($[2, 9]$ and $[3, 11]$) in the persistence diagram that contain the interval $[i, j] = [4, 8]$. The function $(i, j) \mapsto \operatorname{rank}(V(i) \to V(j)) \in \mathbb{N}$, for all choices of $i \leq j$, is called the *rank invariant*.

2.4. **The bottleneck distance.** Let $\operatorname{Dgm}_k(V)$ and $\operatorname{Dgm}_k(W)$ be two persistence diagrams associated with two persistence modules V and W. The distance between two points $x = (x_1, x_2)$ in $\operatorname{Dgm}_k(V)$ and $y = (y_1, y_2)$ in $\operatorname{Dgm}_k(W)$ is given by the L_∞ distance $\|x - y\|_\infty = \max\{|x_1 - y_1|, |x_2 - y_2|\}$.

Definition 2.1. The *bottleneck distance* between the two persistence diagrams $\operatorname{Dgm}_k(V)$ and $\operatorname{Dgm}_k(W)$ is computed by taking the supremum of the L_∞ distance between matched points and then taking the infimum over all bijections $h \colon \operatorname{Dgm}_k(V) \to \operatorname{Dgm}_k(W)$ [26]:

$$d_b(\operatorname{Dgm}_k(V), \operatorname{Dgm}_k(W)) = \inf_h \sup_{v \in \operatorname{Dgm}_k(V)} \|v - h(v)\|_\infty .$$

Note that such a bijection h always exists because we have defined a persistence diagram to contain each point on the diagonal with infinite multiplicity.

Later in the paper, we use the notation $d_b(\operatorname{PH}(\operatorname{VR}(X)), \operatorname{PH}(\operatorname{VR}(Y)))$ to explicitly make clear that the persistence diagrams we refer to are the Vietoris–Rips complexes corresponding to point clouds X and Y. The persistent homology depends on the chosen homological dimension k, but we make this dependency implicit and suppress k from the notation as the statements we consider are often identical for any integer $k \geq 0$.

See Figure 4 for an illustration of the bottleneck distance where the round points correspond to one persistence module and the triangular points correspond to another. The bottleneck distance considers all matchings between the round and triangular points—where unmatched points can be paired with points on the diagonal—and finds the matching that minimizes the largest distance between any two matched points.

3. **Related work.** We now survey related work on vineyards, crocker plots, and other topological and machine learning techniques for studying time-varying metric spaces.

3.1. **Vineyards.** One way to construct a topological summary of a time-varying collection of metric spaces is a *vineyard*. This summary represents a metric space that is varying over time $t \in [0, T]$ as a stacked set of persistence diagrams as time varies, with time-varying curves drawn through the persistence diagram points [19]. A stacked set of persistence diagrams can be thought of as a video for visualization purposes, where each frame corresponds to a 2D persistence diagram, and the persistence diagram points evolve over the time interval $t \in [0, T]$. See for example Figure 5.

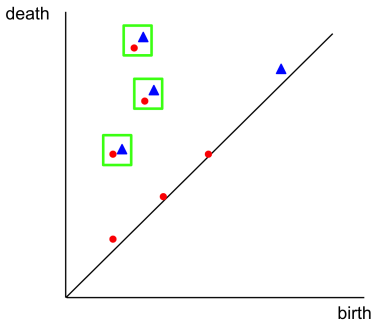


FIGURE 4. Round points (in red) represent points in persistence diagram A. Triangular points (in blue) represent points in persistence diagram B. The bottleneck distance between persistence diagrams A and B is computed by taking the largest L_∞ distance between matched round and triangular points of a bijection between A and B, and then taking the infimum over all bijections. The three boxes (in green) show the optimal matching of three pairs of round and triangular points. The unboxed points match to the closest points on the diagonal.

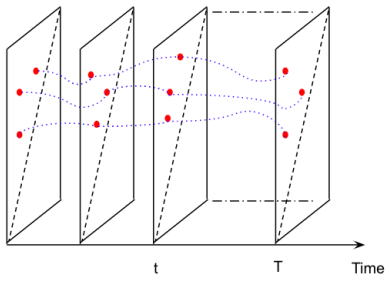


FIGURE 5. Vines and vineyard. Each point (in red) represents a point on a persistence diagram. Each dashed curve (in blue) is a vine traced out by a persistent point on time-varying persistence diagrams. The horizontal direction denotes time.

A vineyard contains a set of curves in 3D, and each curve corresponds to a point in a time-varying persistence diagram in the 3D space $[0, T] \times \mathbb{R}^2$. That is to say, the three coordinates are the time coordinate t, the birth scale of a topological feature, and the death scale of a topological feature. One of the nice properties of persistent homology is that it is *stable* [18, 17], which means that small perturbations in a time-varying metric space lead to small changes in the persistence diagram plots at each point in time.

Thus, when vineyards are considered only as a stacked set of persistence diagrams, they are stable. However, the individual vines are not stable: if two vines move near to each other but then pull apart without touching, so that their persistence diagram points return to their original locations, then after a small perturbation these two vines may instead cross and have their corresponding persistence diagram points switch locations [44]. As such, throughout the paper, we will refer

to a vineyard as a "stacked set of persistence diagrams" or a "vineyard", when the distinction between the two notions matters. More formally, for us a stacked set of persistence diagrams is a continuous map from the interval $[0, T]$ to the space of persistence diagrams equipped with the bottleneck distance. A vineyard furthermore contains the information of time-varying curves of the persistence diagram points.

The paper [19] describes the theory of computing vineyards, which are implemented in, for example, the Dionysus software package [45]. More specifically, we can compute the initial persistence diagram at time $t = 0$, which has a sub-cubic running time in the number of simplices [43], and then update the vineyard as t increases in a linear way (linear in the number of simplices whose orderings in the filtration get transposed).

3.2. **Crocker plots.** Given the same input as a vineyard, *i.e.* a metric space that is varying over time $t \in [0, T]$, the crocker plot gives a topological summary that is an integer-valued function on $\mathbb{R}^2$, where the first input is time t, the second input is the scale ε, and the value of the function is the k-dimensional Betti number of the corresponding Vietoris–Rips complex [55]. This function of two variables (t and ε) can be discretized as a matrix and viewed as a contour diagram; see Figure 9 in Section 5.2.4 as an example. The crocker plot is sometimes better suited for applications than a vineyard, since most scientific images are in 2D as opposed to 3D. A crocker plot is a vector in Euclidean space, and as such, can naturally be used as a feature vector in machine learning tasks as opposed to the non-vector vineyard representation. One drawback of a crocker plot is that it is not stable — perturbing the dynamic metric space only slightly could produce changes of unbounded size in the crocker plot (see Example 2 in Section 7.1).

One can think of the crocker plot as a collapsed or projected version of the vineyard: it is collapsed in the sense that it is lower dimensional (2D instead of 3D), and also in the sense that from a vineyard you can produce the corresponding crocker plot but not vice-versa. As we will show in Section 4.2, if we fix t, the crocker plot recovers the k-dimensional Betti curve.

3.3. **Time-varying metric spaces.** We now review works which analyze the topological structure of time-varying metric spaces.

In the proof-of-concept paper [55], Topaz *et al.* develop the crocker plot and apply it to four realizations of numerical simulations arising from the influential biological aggregation models of Vicsek [57] and D'Orsogna [25]. Traditionally, order parameters derived from physics, such as polarization of group motion, angular momentum, etc., are calculated to assess structural differences in simulations. The authors compare these order parameters to the topological crocker plot approach and discover that the latter reveals dynamic changes of these time-varying systems not captured by the former.

Ulmer *et al.* [56] use crocker plots to analyze the fit of two mathematical, random walk models developed by [47] to experimental data of pea aphid movement. One model incorporates social interaction of the aphids, which is thought to be of importance for pea aphid movement, while the other is a control model. The authors compare time-varying data from the models to time-varying data from the experiments using statistical tests on three metrics (order parameters commonly used in collective motion studies, order parameters that use *a priori* input knowledge of the models, and the topological crocker plots). The topological approach performs as

well as the order parameters that require prior knowledge of the models and better than the ones that do not require prior knowledge, indicating that the topological approach may be useful to adopt when one has little information about underlying model mechanics.

Bhaskar *et al.* [6] use crocker plots coupled with machine learning for parameter recovery in the model of D'Orsogna [25]. The authors generate a large corpus of simulations with varying parameters, which result in different phenotypic patterns of the simulation. For instance, over time, the particles may exhibit the structure of a single or double mill, collectively swarm together, or escape. Each simulation is then transformed into a feature vector that summarizes the dynamics: either the aforementioned time series of order parameters (polarization, angular momentum, absolute angular momentum, average distance to nearest neighbors, or the concatenation of all four) or vectorized crocker plots corresponding to 0-dimensional homology H_0, 1-dimensional homology H_1, or the concatenation of the two. The feature vectors are then fed into both supervised and unsupervised machine learning algorithms in order to deduce phenotypic patterns or underlying parameters. In all cases, the topological approach gives more accurate results than the order parameters, even with no underlying knowledge of the dynamics.

While these papers highlight that the crocker representation can be effective at modeling time-varying data and is amenable as a feature vector for machine learning tasks, crocker plots do not consider the persistence of topological features. At each time step, a crocker plot summarizes only Betti numbers at each scale independently. For example, while a H_1 crocker plot contains the topological information about the number of topological 1-dimensional holes at each scale, it does not encode the information of the scales when holes first appear and subsequently disappear. As a result, crocker plots cannot show if the 1-dimensional holes at two different scales are in fact the same features or different. Also as mentioned above, crocker plots are not stable. In other words, small perturbations in dynamic metric spaces can produce changes of unbounded size in crocker plots, exemplified by Example 2. We address this problem and develop a persistent version of crocker plots, a *crocker stack*, introduced in Section 4, which can capture persistent structural features of time-varying systems.

In contrast to the crocker plot approach, Corcoran *et al.* [20] model swarm behavior of fish by computing persistent topological features using *zigzag persistent homology* [10, 12]. Briefly, a zigzag persistence module connects topological spaces via inclusion maps using either forward or backward arrows and tracks topological features through these inclusions. The authors use inclusions from two consecutive time steps to the union of these time steps to track the evolution of features through time. The resulting persistence diagram is transformed to a *persistence landscape* [7], which is a stable functional representation related to the rank invariant. The persistence landscape exists in a normed vector space, and as such, the authors of [20] use statistical tests to cluster swarm behavior of fish into frequently occurring behaviors named flock, torus, and disordered. However, while this method is persistent in time, to compute zigzag persistent homology for a fixed scale parameter requires *a priori* knowledge of the underlying data to choose an appropriate scale.

In a more theoretical exploration, Kim and Mémoli [36] develop persistent homology summaries of time-varying data by encoding a finite dynamic metric space as a

zigzag persistence module. They prove that the resulting persistence diagram is stable under perturbations of the input dynamic graph in relation to defined distances on the dynamic graphs. In a related research direction, these authors also consider an invariant for dynamic metric spaces in [37]. They introduce a spatiotemporal filtration which can measure subtle differences between pairs of dynamic metric spaces. By producing a 3D persistence module where one of the dimensions is not the real line but a poset, the invariant obtains higher differentiability power than a vineyard representation. Intuitively, [37] considers smoothings in both time and space. By contrast, crocker stacks focus on smoothings in space alone, with the goal of obtaining vectorizable summaries as input for machine learning tasks.

To a lesser degree, our techniques are also related to multiparameter persistence [13, 14, 15, 16, 38, 42, 53] as we compare how time-varying metric spaces evolve in both the parameters of time and scale. However, unlike the scale parameter, there is not inclusion from one time step to the next, and as such, there is not an increasing filtration in time. See also [41] for a functorial model of time, discretized using cellular cosheaves. The papers [1, 3, 11, 22, 23] consider time-varying notions of topology applied to coverage problems in mobile sensor networks.

4. Crocker stacks.

We now describe the crocker plot summary of a time-varying metric space, and its extension to a crocker stack. We give a precise definition of time-varying metric spaces and time-varying persistence modules suitable for our context, describe crocker plots and α-smoothed crocker plots, and finally introduce crocker stacks.

4.1. Time-varying metric spaces and persistence modules.

We use bold letters to denote time-varying objects, and non-bold letters to represent objects at a single point in time.

A *time-varying metic space* $\mathbf{X} = \{X_t\}_{t\in[0,T]}$ is a map $t \mapsto X_t$ from the interval $[0,T] \subseteq \mathbb{R}$ to the collection of all compact metric spaces. Here X_t is a single metric space at the fixed point in time t. We say $\mathbf{X}$ is continuous if this map $t \mapsto X_t$ is continuous with respect to the Gromov–Hausdorff distance (Section 6.2), and we say $\mathbf{X}$ is finite if there is some integer N such that the cardinality of metric space X_t is at most N for all $t \in [0,T]$.

For example, if Z is a fixed metric space (perhaps Euclidean space $Z = \mathbb{R}^n$), and if $x_1, \ldots, x_N \colon [0,T] \to Z$ are a collection of N continuous maps into Z, then we can form a time-varying metric space $\mathbf{X} = \{X_t\}_{t\in[0,T]}$ by letting $X_t = \{x_1(t), \ldots, x_N(t)\}$. We note that $\mathbf{X}$ is both continuous and finite. Many of the time-varying metric spaces that we consider when studying agent-based collective motion models are constructed in this way.

Similarly, a *time-varying persistence module* $\mathbf{V} = \{V_t\}_{t\in[0,T]}$ is a map $t \mapsto V_t$ from the interval $[0,T] \subseteq \mathbb{R}$ to the collection of all persistence modules, equipped with the bottleneck distance. Here V_t is a single persistence module at the fixed point in time t. We say that $\mathbf{V}$ is continous if this map $t \mapsto V_t$ is continuous.

If $\mathbf{X}$ is a time-varying metric space, then we can form a time-varying persistence module $\mathbf{V} = \mathrm{PH}(\mathrm{VR}(\mathbf{X}))$ defined for each $t \in [0,T]$ by $V_t = \mathrm{PH}(\mathrm{VR}(X_t))$. Here we have fixed the homological dimension $k \geq 0$ and supressed it from the notation. It follows from the stability of persistent homology (Section 6.3) that if $\mathbf{X}$ is continuous, then so is $\mathbf{V}$.

We remark that a vineyard contains more information than a time-varying persistence module. Indeed, a vineyard additionally contains a matching between the points in the persistence diagram V_t and $V_{t+\varepsilon}$ for $\varepsilon > 0$ sufficiently small.

4.2. **Crocker plots.** Crocker plots were originally defined on time-varying metric spaces $\mathbf{X}$, after first applying Vietoris–Rips complexes and then homology to get the time-varying persistence module $\mathbf{V} = \{V_t\}_{t\in[0,T]}$, where $V_t = H_k(\mathrm{VR}(X_t;\varepsilon))$ [55]. Here, we take the more general approach and define a crocker plot for any time-varying persistence module $\mathbf{V}$, regardless of its origins. In fact, we note a crocker plot can be formed for any two parameter family of topological spaces, such as a bifiltration, not merely one varying in time. Recall that a higher-dimensional analogue of a matrix is called a tensor. More generally, given an m-parameter family of topological spaces, one can create an m-dimensional tensor analogue of a crocker plot that contains a Betti number at each entry in the tensor, though we do not pursue that topic here.

Let $\mathbf{V}$ be a time-varying persistence module, with V_t the persistence module at time t. In the 2D *crocker plot* of homological dimension k, the value at time t and scale parameter ε is the rank (or dimension) of the vector space $V_t(\varepsilon)$. This function of two variables can be viewed as a contour plot, as shown in Figure 9, which is suitable for applications as all times are displayed simultaneously. For discretized values of ε and t (as for computational purposes), the ranks of $V_t(\varepsilon)$ can be represented as a matrix. This matrix can then be vectorized and used as a feature vector for machine learning algorithms.

If V_t is the k-dimensional homology of the Vietoris–Rips complex of a metric space X_t, then taking this time-varying metric space as input, we obtain a crocker plot, which again returns a topological summary that is an integer-valued function on $\mathbb{R}^2$. The rank at time $t \in [0,T]$ and scale ε is then the number of "k-dimensional holes at scale ε", also known as the Betti number β_k. For a fixed t and varying ε, this is equivalent to the notion of a *Betti curve*.

4.3. **α-smoothed crocker plots.** An extension of a crocker plot is an *α-smoothed* crocker plot. When applied to a time-varying persistence module $\mathbf{V} = \{V_t\}_{t\in[0,T]}$, the output of an α-smoothed crocker plot for $\alpha \geq 0$ is the rank of the map $V_t(\varepsilon - \alpha) \to V_t(\varepsilon + \alpha)$ at time t and scale ε. A standard crocker plot can also be called a *0-smoothed* crocker plot. The effect of α-smoothing is shown in Figure 6. Note that α-smoothing can potentially reduce noise in a crocker plot.

4.4. **The crocker stack for time-varying persistence diagrams.** The *crocker stack* is a sequence, thought of as a video, of α-smoothed crocker plots, in which each α-smoothed crocker plot is a frame of the video for continuously increasing α, starting at $\alpha = 0$.

Definition 4.1. A *crocker stack* summarizes the topological information of a time-varying persistence module $\mathbf{V}$ in a function $f_{\mathbf{V}} \colon [0,T] \times [0,\infty) \times [0,\infty) \to \mathbb{N}$, where

$$f_{\mathbf{V}}(t,\varepsilon,\alpha) = \mathrm{rank}(V_t(\varepsilon - \alpha) \to V_t(\varepsilon + \alpha)).$$

In the 3D domain $[0,T] \times [0,\infty) \times [0,\infty)$, the horizontal axis displays the time $t \in [0,T]$, and the vertical axis displays the scale $\varepsilon \in [0,\infty)$. The persistence parameter $\alpha \in [0,\infty)$ indicates the order of the video frames. In other words, the crocker stack is a sequence of α-smoothed crocker plots that vary over $\alpha \geq 0$. As examples, see Figures 2 and 10.

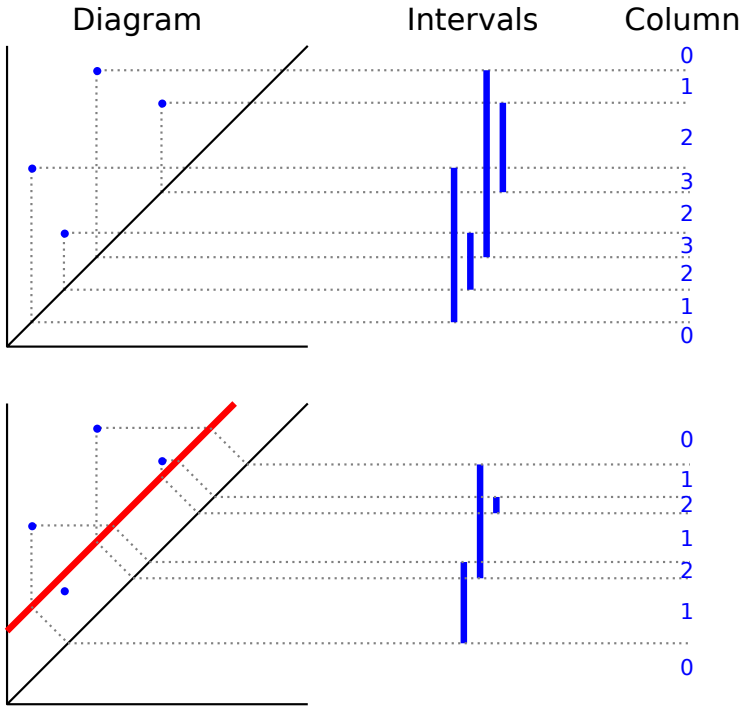


FIGURE 6. The effect of α-smoothing. (Top) A persistence diagram, the corresponding persistence intervals (drawn vertically), and one column of a crocker plot matrix. If we had points moving in time, then we would get a time-varying persistence diagram, a time-varying persistence barcode, and a complete crocker plot matrix (swept out from left to right as time increases). (Bottom) A persistence diagram with the thick line (in red) reflecting the choice of α-smoothing, along with the corresponding α-smoothed persistence intervals, and one column of an α-smoothed crocker plot matrix. The y-intercept of the diagonal thick red line is 2α. All persistence diagram points under the thick line are ignored under α-smoothing.

The crocker stack has the property that $f_{\mathbf{V}}(t, \varepsilon, \alpha) \leq f_{\mathbf{V}}(t, \varepsilon, \alpha')$ for $\alpha \geq \alpha'$: larger α values require features to persist longer, which means that the crocker stack is a non-increasing function of α. Viewing a crocker stack as α increases could help one identify interesting smoothing parameter choices α to consider.

In Section 6.5, we describe how the crocker stack and the stacked set of persistence diagrams are equivalent to one another, in the sense that either one contains the information needed to reconstruct the other. However, crocker stacks can sometimes display the equivalent information in a more useful format. This is because in a crocker stack, all times $t \in [0, T]$ are represented in each frame α, whereas in a time-varying persistence diagram during a single frame t one only ever sees information about that time. We also explain in Section 7 how crocker stacks are continuous, though, unfortunately, not in a way that would be the most immediately pertinent for machine learning applications.

5. Experiments with the Vicsek model.

5.1. **Background.** We now turn to evaluating the utility of α-smoothed crocker plots for applications. This assessment centers on a parameter identification task for a mathematical model developed by Vicsek and collaborators [57]. *Parameter identification* refers to deducing the parameters of a model from experimental or simulation data. The *Vicsek model* is a seminal model for collective motion, in which agents attempt to align their motion with that of nearby neighbors, subject to a bit of random noise. Concretely, the question we ask is "given time series data output from the Vicsek model, can we recover the model parameters responsible for simulating that data?"

The significance of this parameter identification task stems from the importance of the Vicsek model itself, which is arguably one of the most widely-adopted models in the study of active matter. As of the time of the drafting of this manuscript, [57] has been cited nearly 7,000 times. Originally motivated by the collective motion of fish, birds, insects, and mammals, the model has also been used or adapted to describe other types of interacting agents, including vibrating rods, actin filaments, bacterial colonies, skin pigment cells, and more [58]. It also is closely related to ferromagnetic behavior, to the liquid-gas transition, and other phenomena. It is a model that is both simple and universal. The model has three effective parameters: population density, agent speed, and agent noise. In biological settings, one can definitively know the population density (by counting agents). The agent speed is rather straightforward to measure using motion tracking. The most elusive parameter is the noise parameter, which there is no direct way to measure as it is an inherent behavioral property of the agent. As such, we study the noise parameter in the Vicsek model to demonstrate a way to overcome a challenge associated with an extremely influential model.

More specifically, the noise parameter, η, in the Viscek model measures the degree of randomness in an agent's chosen direction of motion. We generate 100 simulations for each of 15 different values of η, for a total of 1500 simulations. We then create four different experiments, each using simulation data from a chosen subset of η values.

An experiment consists of the following procedure. For all of the simulations admitted to the experiment, we compute time series of feature vectors that summarize the simulation data: an order parameter from the physics literature that measures alignment of agents, α-smoothed crocker plots, a (discretized) crocker stack, and a stacked set of persistence diagrams (in one experiment). As a reminder, the terminology "stacked set of persistence diagrams" is used in place of a vineyard when we do not consider the vines or curves traced out by the persistence points. For the first three feature vectors, we compute pairwise distances between simulations using a Euclidean norm, and for the stacked set of persistence diagrams, we compute (the computationally expensive) supremum bottleneck distance. These pairwise distances are the inputs to a machine learning algorithm, K-medoids clustering. We then assess the success of clustering under different choices for the distance metric by examining how many simulations were clustered to a medoid with the same η value.

For the remainder of this section, we will present the Vicsek model, describe the design of our numerical experiments, and present the results of using various metrics for parameter identification methods. These results allow us to compare the efficacy of topological approaches to that of more traditional ones.

5.2. **Research design.**

5.2.1. *Vicsek model.* Cited thousands of times in the scientific literature, the Vicsek model is a seminal model for collective motion [57]. This discrete time, agent-based model tracks the positions $\vec{x}_i$ and headings $\theta_i \in [0, 2\pi)$ of n agents in a square region with periodic boundary conditions. The agents are seeded with uniformly random positions and uniformly random initial headings. At each time step, an agent updates its heading and its position. The new heading is taken to be the average heading of nearby neighbors, added to a small amount of random noise drawn from the uniform distribution $(-\eta/2, \eta/2)$. More explicitly, the model is:

$$\theta_i(t + \Delta t) = \frac{1}{N}\left(\sum_{|\mathbf{x}_i - \mathbf{x}_j| \le R} \theta_j(t)\right) + U(-\tfrac{\eta}{2}, \tfrac{\eta}{2}).$$

Above, R is the distance threshold under which nearby neighbors interact, N is the number of neighbors within distance R, and U is the uniform distribution. See Figure 7 for an illustration of this model.

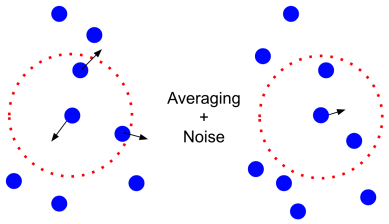


FIGURE 7. To update the heading of an agent (centered, round blue point) according to the Vicsek model, we first find the nearby neighbors within a radius R (denoted by the dashed circle in red) and then take the average of its neighbors' headings, plus some noise.

The Vicsek model updates θ_i and $\vec{x}_i$ according to an Euler's method type of update, that is,

$$\mathbf{v}_i(t + \Delta t) = v_0(\cos\theta_i(t + \Delta t), \sin\theta_i(t + \Delta t))$$

$$\mathbf{x}_i(t + \Delta t) = \mathbf{x}_i(t) + \mathbf{v}_i(t + \Delta t)\Delta t.$$

Note that all particles move with the same constant speed v_0.

5.2.2. *Parameters and simulations.* The parameters in the Vicsek model are the number of agents n, the radius of alignment interaction R, the level of noise η, the length of a side of the periodic domain ℓ, the agent speed v_0, and the time step Δt. In the literature on this model, Δt and R are typically set to unity without loss of generality since one may rescale time and space. This leaves the parameters n, η, ℓ, and v_0. Due to the periodic domain and finite sensing radius of the agents, these collapse effectively into three parameters, namely v_0, η, and an agent density $\rho = n/\ell^2$.

For each simulation, we fix length $\ell = 25$ and $n = 300$ agents, which gives that $\rho = 0.48$. We also take speed $v_0 = 0.03$. The remaining noise parameter η is the

value that we hope to predict from the output of the simulation by using machine learning. We generate 100 simulations for each of 15 η values:

$$\eta \in \{0.01, 0.02, 0.03, 0.05, 0.1, 0.19, 0.2, 0.21, 0.3, 0.5, 1, 1.5, 1.9, 1.99, 2\}.$$

We compare four methods of predicting η: order parameters, as described in Section 5.2.3, crocker plots and stacks, as introduced in Section 5.2.4, and in the case of one experiment, stacked sets of persistence diagrams, as described in Section 3.1.

Simulation data sets generated from the Vicsek model with different noise parameters η include four variables: time t, position coordinates x and y, and heading θ. We then transform the heading θ to velocity $\mathbf{v} = (v_x, v_y)$. This velocity can be used for computing the alignment order parameter for each simulation at each time step t, defined in Section 5.2.3.

To compute the persistent homology of simulations for crocker plots, discussed in more detail in Section 5.2.4, we use both the 2-dimensional position data and the heading of each agent at each time step. The x and y positions are scaled by $1/\ell = 1/25$, the length of the box, and the heading is scaled by $1/2\pi$, in order for all data to be in the range from 0 to 1.

In Section 5.2.5, we will design four machine learning experiments by including different combinations of the 15 η noise values. We choose time steps 1, 10, and 40 for different experiments and will show the effect of the size of time steps on clustering accuracy.

Vicsek's original work [57] identifies several different possible qualitative behaviors of the system. To quote him directly, "For small densities and noise, the particles tend to form groups moving coherently in random directions... at higher densities and noise, the particles move randomly with some correlation. For higher density and small noise, the motion becomes ordered."

5.2.3. *Alignment order parameter.* Order parameters are a common way to measure the level of global synchrony in an agent-based model over time. The alignment order parameter

$$\varphi(t) = \frac{1}{nv_0} \left\| \sum_{i=1}^{n} \mathbf{v}_i(t) \right\|$$

is defined as the normalized magnitude of the average of the velocity vectors at time t, where n is the number of particles in the model, as above. This creates a time series recording the degree to which particles are aligned at each time, with 1 indicating a high degree of alignment and 0 indicating no alignment. Calculating the order parameter is a conventional approach derived from physics. We want to compare this approach to topological approaches on the task of parameter identification, by clustering simulations of biological aggregations with the Vicsek model into corresponding groups based on the noise parameter.

Figure 8 displays the alignment order parameters for three simulations with different noise parameters $\eta = 0.01$, 1, 2, and their changes over time. At time zero, all three values of the order parameter are quite low due to the random initialization of headings. All three simulations exhibit some alignment over time, with the smaller noise parameters reflecting a higher level of synchrony, consistent with Vicsek's description above.

5.2.4. *Crockers.* Crocker plots, α-smoothed crocker plots, and crocker stacks (collectively referred to as crockers) are three inputs that we will test in our machine

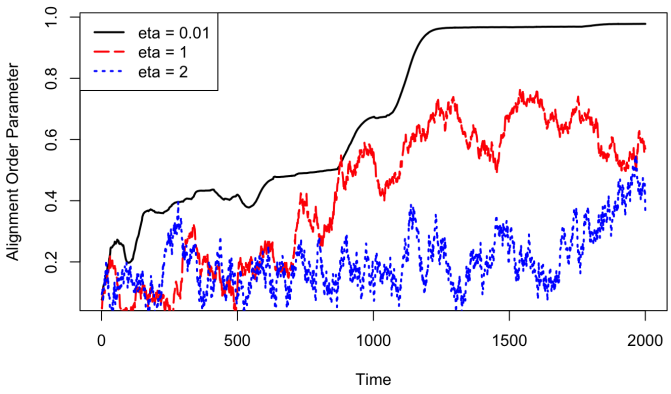


FIGURE 8. A plot of order parameters for three simulations of the Vicsek model with different noise parameters η. For smaller values of η, particles become more aligned, *i.e.* move in the same direction, over time.

learning experiments. We compute persistent homology using the R software package "TDA" [28], subsampling to every 10 time steps to speed up computations. At each subsampled time, we compute the persistent homology of Vietoris–Rips filtered simplicial complexes, with vertex set given by the scaled location and heading of each agent. We then compute the crockers from these persistent homology intervals over all times.

We make the following choices in the computations of our crockers.

- We only compute persistent homology in dimensions zero and one.
- We compute the Vietoris–Rips filtration up to scale parameter $\varepsilon = 0.35$. These computations are discretized to consider 50 equally-spaced values of ε between $\varepsilon = 0$ and $\varepsilon = 0.35$, inclusive.
- We consider 18 different smoothing values: from $\alpha = 0$ to $\alpha = 0.17$ inclusive, with steps of size 0.01 in between.

Figure 9 is an example of an H_0 crocker plot for a simulation with noise parameter $\eta = 0.02$. We display only the contours of Betti numbers $\beta_0 \leq 5$, interpreting larger contours as noise. In Figure 9, notice the large region with two connected components, namely $\beta_0 = 2$, over the time range from approximately $t = 1100$ to 2000. This can be interpreted as two connected components for a wide range of both scale parameter ε and simulation time t. By looking only at the 0-smoothed crocker plot, there is *a priori* no guarantee that these are the *same* components as scale ε varies. However, as we will show in Section 6, since the crocker stack contains enough information to recover the persistent homology barcodes, one can confirm these are the same connected components as ε varies by considering the later smoothings $\alpha > 0$ in the crocker stack. To verify these are the same connected components also as time t varies, one would instead want to look at the vineyard representation.

Figure 10 illustrates a stack of α-smoothed H_0 and H_1 crocker plots for the same simulation, with the α values 0, 0.01, and 0.03 shown. In H_0, the Betti curves translate down as α increases; by contrast, in H_1, the Betti curves morph shapes as α increases.

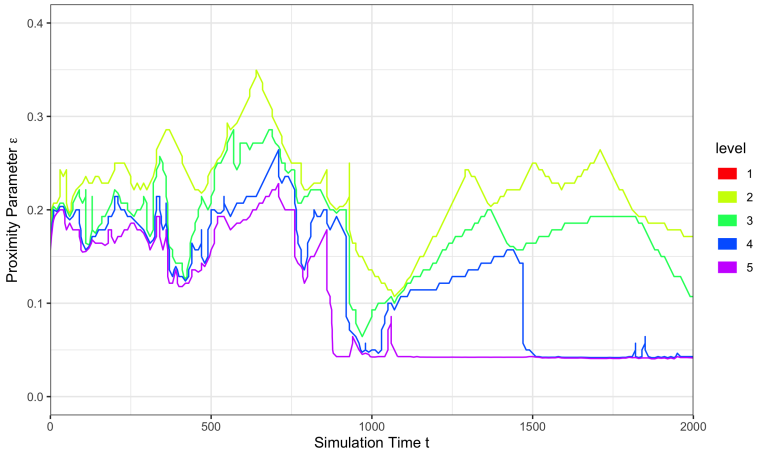


FIGURE 9. An example H_0 crocker plot of a simulation from the Viscek model with noise parameter $\eta = 0.02$. This is the same as an α-cross section of a crocker stack when $\alpha = 0$. In the region below the lowest curve (purple) where $\beta_0 \geq 5$, there can be many connected components, which we interpret as noise and is thus not displayed.

We will cluster different simulations based on their crocker representations, including crocker plots, single α-smoothed crocker plots, and a crocker stack, and we compare the clustering accuracies in Section 5.3.

5.2.5. *Experiments.* We create four different experiments of increasing levels of clustering difficulty by considering different collections of noise values η that we will try to predict from simulated data.

- Experiment 1. Five η values: $\eta = 0.01, 0.5, 1, 1.5, 2$.
- Experiment 2. Three η values: $\eta = 0.01, 0.1, 1$.
- Experiment 3. Six η values: $\eta = 0.01, 0.02, 0.19, 0.2, 1.99, 2$.
- Experiment 4. Fifteen η values: $\eta = 0.01, 0.02, 0.03, 0.05, 0.1, 0.19, 0.2, 0.21, 0.3, 0.5, 1, 1.5, 1.9, 1.99, 2$.

Note that the more similar η is, the more difficult it will be to cluster the data correctly. As a thought experiment, consider two simulations with $\eta = 0.01$ and 0.02, respectively. Both simulations will very quickly produce an aligned group, and once the groups are aligned, they are nearly indistinguishable. The most distinguishing data is during the transient times, but due to the quick equilibration of the system, there is relatively little transient data.

In all four experiments, we use time step equal to 10 for the crockers; additionally, we compute the crocker plot for time step equal to 40 in Experiment 2 to see how subsampling time affects accuracy. When computing the order parameter, we instead use time step equal to 1 (in order to get the "best" result the order parameter can provide); in Experiment 2, we also compute the order parameter with time step equal to 10 and 40 for comparison purposes.

5.2.6. *Distance matrices.* In each experiment, we vectorize the order parameter time series and crocker representations for each simulation, calculate the pairwise Euclidean distance between those vectors, and summarize the distances in a pairwise

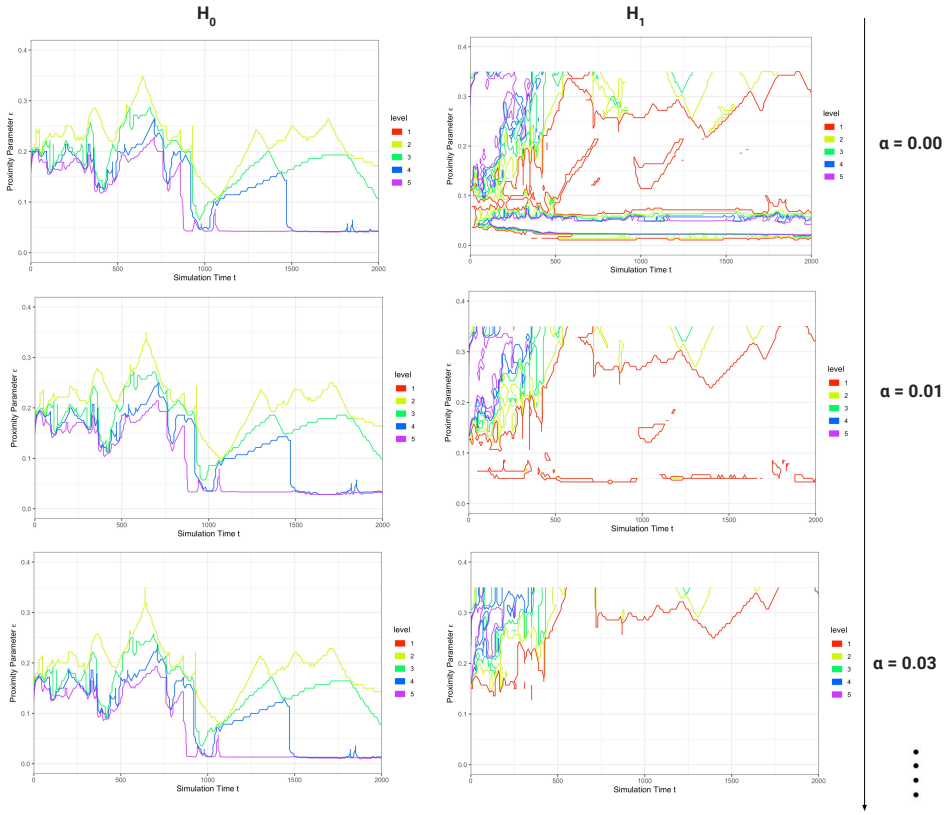


FIGURE 10. An example H_0 and H_1 crocker stack for a simulation from the Viscek model with noise parameter $\eta = 0.02$. This figure shows the shifts of Betti curves in H_0 and H_1 as smoothing parameter α increases from 0 to 0.01 and 0.03.

distance matrix. To vectorize the crocker representations, we take the crocker matrix (or α-smoothed crocker matrix) and concatenate the rows, and for the crocker stack, we further concatenate each level of α. Images of distance matrices for the order parameter and $H_{0,1}$ crocker plot (where the H_0 and H_1 vectorized crocker plots have been concatenated) are shown in Figures 11–14 for each of the four experiments, respectively. The distance matrices for crocker stacks are visually similar to those for crocker plots and thus will not be shown here. In each of the four pairs of comparisons, the crocker distance matrix is more structured than the order parameter distance matrix. This is consistent with the higher clustering accuracy based on crocker plots compared to order parameters, which we discuss in Section 5.3. That is, distances between simulations arising from the same (or similar) noise parameter(s) typically have a smaller distance than those from different noise parameters, resulting in a block structure in the crocker distance matrix that is not as apparent in the order parameter distance matrix.

5.2.7. *Clustering method: K-medoids.* We use the *K-medoids* algorithm to cluster the simulations from each experiment [35, 50]. This is an unsupervised clustering

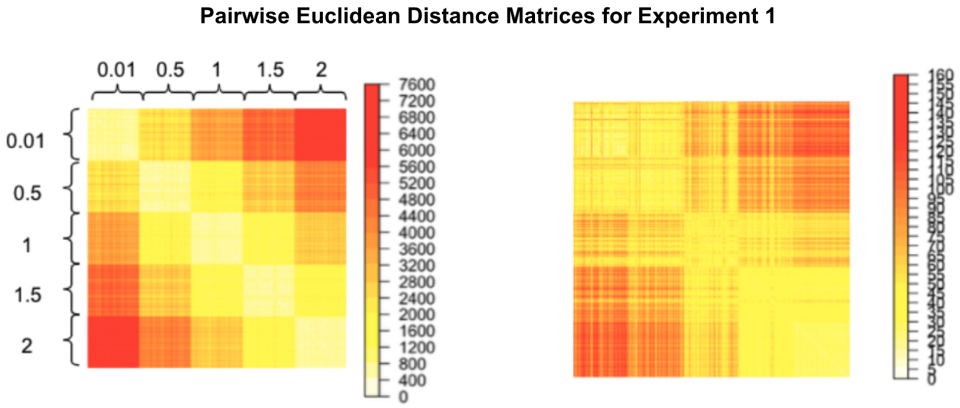


FIGURE 11. The color scale corresponds to values in the distance matrix; denser color (red) means larger distances, and lighter color (yellow) means smaller distances. The 100 simulations of each noise parameter $\eta = 0.01, 0.5, 1, 1.5, 2$ are listed in order and annotated in the left matrix. The $H_{0,1}$ crocker distance matrix is more structured (Left) than the order parameter distance matrix (Right).

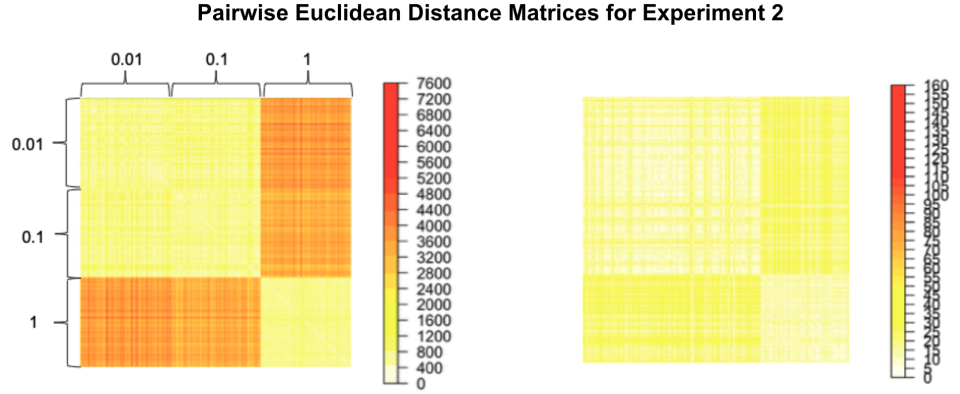


FIGURE 12. The $H_{0,1}$ crocker distance matrix is more structured (Left) than the order parameter distance matrix (Right).

algorithm, meaning that no labels are used to determine the clusters. The algorithm minimizes the sum of pairwise dissimilarities between data points by searching for K objects from the data set, called *medoids*, and then partitioning the remaining observations to their closest medoid, resulting in K clusters. The medoid is the most centrally located object of each cluster and is, in fact, an observation from the data set. Specifically, a medoid will correspond to a particular simulation in our clustering experiments. Post clustering, we can trace each medoid to the underlying noise parameter η of the simulation. The K-medoids algorithm is often more robust than the ubiquitous K-means algorithm, which minimizes the sum of squared Euclidean distance and thus, is more sensitive to outliers [4]. Another benefit of K-medoids is that it can take as input a distance matrix, rather than a set of feature

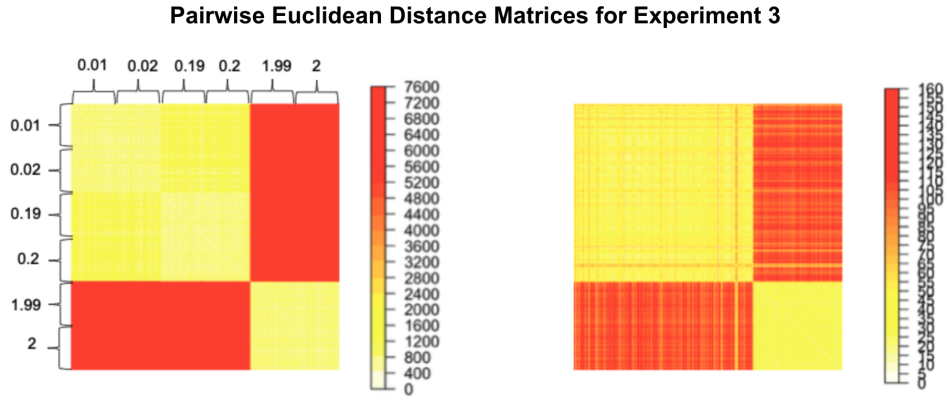


FIGURE 13. The $H_{0,1}$ crocker distance matrix is more structured (Left) than the order parameter distance matrix (Right).

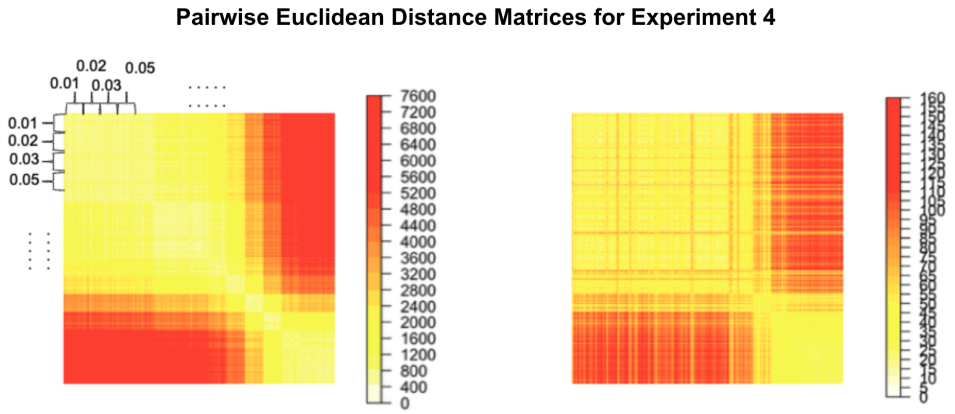


FIGURE 14. The $H_{0,1}$ crocker distance matrix is more structured (Left) than the order parameter distance matrix (Right).

vectors, which will be important in Section 5.3.3 where we compute a distance on stacked sets of persistence diagrams, which do not have vector representations.

We use the *Partitioning Around Medoids (PAM)* algorithm in the R package "cluster" [39] to perform the K-medoids algorithm, setting K to be equal to the number of distinct noise parameter values η in each experiment. For instance, $K = 5$ in Experiment 1, and $K = 15$ in Experiment 4. We compute the classification accuracy as the percentage of simulations found to be in a cluster whose medoid comes from the same noise parameter value, where labels are used post-clustering to determine this accuracy. Simulations partitioned into a cluster whose medoid does not come from the same noise parameter are misclassified.

5.2.8. *PCA vs. non-PCA.* The vectorized version of the alignment order parameter has 2001 time values ($t = 0$ to 2000), and accordingly the time series are of dimension 2001. On the other hand, a crocker plot, which has been downsampled by a factor of 10, has 201 time values and 50 ε values, resulting in a 10050-dimensional vectorized crocker for H_0 and H_1, and resulting in a 20100-dimensional vectorized crocker for

the concatenation $H_{0,1}$. It might not be "fair" to directly compare the clustering results based on the different kinds of vectors of vastly differing dimensions. To address this problem, we reduce the dimension of each vector to three using *Principal Component Analysis (PCA)* [33]. This is reasonable as the first three principal components of each kind of feature vector capture about 90% of the variance. After reducing both the order parameter and crocker vectors for each simulation to 3-dimensional vectors, we create distance matrices based on the PCA-reduced vectors for each experiment, and then compare the PCA clustering accuracy with non-PCA clustering accuracy.

5.3. **Key findings.**

5.3.1. *Results summary.* Table 1 shows a summary of the accuracies from clustering using K-medoids on each of the four experiments with different input feature vectors (order parameters, crocker plots, and crocker stacks) using both the full representations (non-PCA) and dimensionality reduced versions (PCA, italicized). When we refer to the crocker stacks in the table, we are considering the stack of the 18 α-smoothed crocker plots discussed in Section 5.2.4, where we vectorize the crocker for each α and then concatenate. In Section 5.3.2, we consider the clustering accuracy of single α-smoothed crocker plots in comparison with this stacked version. The crocker representations consider the homology dimensions H_0 and H_1 as well as the concatenation $H_{0,1}$ of the two.

TABLE 1. Summary of the clustering accuracy on four different experiments (abbreviated Exp.) with three different feature vectors: order parameters, crocker plots, and crocker stacks. For crocker plots and crocker stacks, we distinguish different homological dimensions: $H_{0,1}$, H_0, and H_1. The top accuracy scores of each column are bolded. This table summarizes results with time step 1 for order parameters and time step 10 for crocker representations. Results with other time steps are discussed in Section 5.3.1. Clustering results with feature vectors that have been reduced to 3 dimensionsby PCA are shown in italics, while full feature vectors are not italicized.

	Exp. 1		**Exp. 2**		**Exp. 3**		**Exp. 4**	
Order Parameters	0.63	*0.51*	0.61	*0.59*	0.35	*0.35*	0.21	*0.17*
Crocker Plots, $H_{0,1}$	**1.00**	***1.00***	0.67	*0.67*	0.44	*0.43*	**0.42**	***0.43***
Crocker Plots, H_0	**1.00**	***1.00***	0.67	***0.77***	0.45	*0.43*	0.39	***0.43***
Crocker Plots, H_1	0.98	*0.99*	**0.71**	*0.67*	0.36	*0.35*	0.37	*0.33*
Crocker Stacks, $H_{0,1}$	**1.00**	*0.98*	0.67	*0.67*	0.47	*0.38*	0.41	*0.35*
Crocker Stacks, H_0	**1.00**	***1.00***	0.67	*0.67*	**0.49**	***0.46***	0.41	*0.41*
Crocker Stacks, H_1	0.96	*0.98*	0.63	*0.67*	0.34	*0.37*	0.32	*0.35*

Crocker plot and stack clustering accuracies are higher than order parameter accuracies across all experiments. We perform paired sample t-tests to compare the means of experiment accuracies between order parameter and crocker plot ($H_{0,1}$)

feature vectors and between order parameter and crocker stack ($H_{0,1}$) feature vectors. The 1-tail p-value for the t-test between order parameters and crocker plots is 0.04, and the p-value for the t-test between order parameters and crocker stacks is 0.03, which are both significant at the significance criterion 0.05 level. In other words, the means of the four experiments' accuracies for both crocker plots and stacks are significantly higher than for order parameters. As expected, Experiment 1 accuracy is the highest compared to other experiments for both order parameters and crocker plots. This is because the noise η parameters are approximately evenly spaced in Experiment 1. Experiment 4 accuracy is the lowest, since Experiment 4 has both close and very different η parameters, which confuses classification. Generally speaking, PCA (shown in italics in Table 1) and non-PCA accuracies are comparable across all four experiments. It is striking to us that even though we have reduced the dimension of the data down to 3 via PCA, we observe little degradation in cluster accuracy. The accuracies for crocker stacks are comparable with those for crocker plots.

Since $H_{0,1}$ distance matrices generally contain more information than either H_0 and H_1, they often generate comparable or higher clustering accuracies than either H_0 or H_1. We observe two exceptions for non-dimensionality reduced crocker plots: H_1 accuracy for Experiment 2 is slightly higher than $H_{0,1}$ and H_0 accuracy; H_0 accuracy for Experiment 3 is slightly higher than $H_{0,1}$ and H_1 accuracy. For non-dimensionality reduced crocker stacks, H_0 accuracy is slightly higher than $H_{0,1}$ in Experiment 3.

Among order parameters, crocker plots, and crocker stacks, the stacks encode the most information, since they contain α-smoothed crocker plots over several different α values. In H_0, the contour lines of α-smoothed crocker plots continuously shift down as α increases. This is because, in a Vietoris-Rips complex, all vertices are born at scale $\varepsilon = 0$. Thus, the contour lines separating different ranks in the crocker plot translate as α increases, but are otherwise unchanged. In contrast, in H_1, the contour lines of α-smoothed crocker plots do not continuously shift down but morph in shape. See the links for videos of H_0 and H_1 crocker stacks from a simulation corresponding to noise parameter $\eta = 0.02$ of the Viscek model.

To further detail how the clustering accuracies in Table 1 arise, we consider Experiment 3 as a case study. Misclassifications can occur for two reasons. First, as alluded to previously, it will be very difficult to accurately cluster simulations that have different values of η that are all in the small-noise regime, leading to strong alignment of the system. When the system aligns strongly and quickly, information contained in the transient state is lost. Second, even when the system is not in the strong-alignment regime, misclassifications may occur simply when values of η are sufficiently close together. These difficulties are exemplified in Table 2, which shows the confusion matrix of the $H_{0,1}$ crocker plots of Experiment 3 clustered using K-medoids, which contains simulations with $\eta = 0.01, 0.02, 0.19, 0.2, 1.99, 2$. The rows represent the actual simulations corresponding to each noise parameter η while the columns represent the parameter of the cluster medoid to which a simulation is assigned. Even though $K = 6$ clusters were formed, the six medoids correspond to simulations from only three distinct noise parameters $\eta = 0.02, 0.2, 2$. That is, there were two medoids from each of these three noise parameter classes selected by the algorithm, and we group clusters with the same noise parameter together. All simulations corresponding to noise parameters $\eta = 0.01, 0.19, 1.99$ are misclassified as there are no medoids selected from these parameter classes. Notice that 69 of 100

TABLE 2. Confusion matrix using K-medoids to cluster the $H_{0,1}$ crocker plots corresponding to simulations of Experiment 3 (η=0.01, 0.02, 0.19, 0.2, 1.99, 2). The rows represent the actual simulations corresponding to each noise parameter η while the columns represent the parameter of the cluster medoid to which a simulation is assigned. Even though $K = 6$ clusters were formed, the six medoids correspond to simulations from only three distinct noise parameters $\eta = 0.02, 0.2, 2$.

	K-medoids Clusters		
	η **= 0.02**	η **= 0.20**	η **= 2.00**
η **= 0.01**	69	31	0
η **= 0.02**	64	36	0
η **= 0.19**	4	96	99
η **= 0.2**	1	99	0
η **= 1.99**	0	0	100
η **= 2.00**	0	0	100

simulations from class $\eta = 0.01$ are partitioned into a cluster with a medoid coming from class $\eta = 0.02$, and 64 of 100 simulations from class $\eta = 0.02$ are partitioned into the same cluster. This pattern is consistent for simulations with η values of the same order of magnitude: simulations from classes $\eta = 0.19$ and 0.2 are largely partitioned into a cluster with a medoid coming from class $\eta = 0.2$, and simulations from classes $\eta = 1.99$ and 2 are all partitioned into a cluster with a medoid coming from class $\eta = 2$. In addition, simulations from the first four classes ($\eta = 0.01$, 0.02, 0.19, 0.2) are partitioned into either of the clusters with medoids from $\eta = 0.02$ or 0.2. Recall that values of η that are in the alignment regime will produce data that is essentially identical apart from a very brief transient phase and hence, are difficult to distinguish.

We now consider the effect of subsampling time on classification accuracy. The order parameter clustering accuracy for Experiment 2 with time step 1 is 0.61, and this is the same as subsampling the data by time steps of 10 and 40. In addition, the crocker plot and crocker stack accuracy for Experiment 2 with time steps 10 and 40 are the same. This shows that the effect of these subsamplings of time on clustering accuracy is negligible in this particular experiment.

5.3.2. *$\alpha-$smoothed crocker plots: Stack vs. single α's.* We compare the clustering accuracy of the $H_{0,1}$ stacked α-smoothed crocker plots (with 18 α values combined) and those with a single α value in Table 3. The crocker stack performs comparably to the individual α-smoothed crocker plots.

As α increases, accuracy decreases. This makes sense because as α increases, more smoothing of distinct topological features occurs, which may ignore some information necessary for parameter identification.

5.3.3. *Distances on stacked sets of persistence diagrams.* We now want to compare the clustering accuracy of the crocker plot representations to the stacked set of persistence diagrams representation. Recall as introduced in Section 3.1, the latter

TABLE 3. Summary of the clustering accuracy for the four experiments based on single α-smoothed crocker plots in $H_{0,1}$ as input feature vectors to K-medoids, and the accuracy based on the crocker stack (with 18 α values combined). The top accuracy scores of each column are bolded. Recall that when $\alpha = 0$, the α-smoothed crocker plot is equivalent to the standard crocker plot of [55].

	Exp. 1	**Exp. 2**	**Exp. 3**	**Exp. 4**
stack	**1.00**	0.67	0.47	0.41
$\alpha =$ **0.00**	**1.00**	0.67	0.44	**0.42**
$\alpha =$ **0.01**	**1.00**	0.67	0.46	0.40
$\alpha =$ **0.03**	**1.00**	0.67	**0.49**	0.40
$\alpha =$ **0.05**	**1.00**	0.58	0.44	0.38
$\alpha =$ **0.08**	0.99	0.67	0.39	0.36
$\alpha =$ **0.11**	0.97	0.67	0.33	0.35
$\alpha =$ **0.13**	0.92	**0.73**	0.41	0.28
$\alpha =$ **0.17**	0.73	0.66	0.40	0.21

can be thought of as time-varying persistence diagrams, since they contain the births and deaths of topological features over the scale parameter ε and time t. Suppose two time-varying metric spaces $\mathbf{X}$ and $\mathbf{Y}$ have persistence diagrams $PH(\mathrm{VR}(X_t))$ and $PH(\mathrm{VR}(Y_t))$ for all t. One way to compute distance between these stacked sets of persistence diagrams is to compute the bottleneck distance between each pair of persistence diagrams $PH(\mathrm{VR}(X_t))$ and $PH(\mathrm{VR}(Y_t))$ at each time t and then take the supremum of the bottleneck distances over all time values. This *supremum bottleneck distance* (which we henceforth refer to as the bottleneck distance between stacked sets of persistence diagrams) is defined as follows:

$$d_b^{\infty}(\mathrm{PH}(\mathrm{VR}(\mathbf{X})), \mathrm{PH}(\mathrm{VR}(\mathbf{Y}))) = \sup_t d_b\Big(\mathrm{PH}(\mathrm{VR}(X_t)), \mathrm{PH}(\mathrm{VR}(Y_t))\Big).$$

As this computation is extremely expensive, we only compute this bottleneck distance for Experiment 2, which entails 300 simulations. The K-medoids clustering accuracies based on this bottleneck distance matrix, along with corresponding accuracies of the Euclidean distances of the order parameters, crocker plots, and crocker stacks, are shown in Table 4.

As a stacked set of persistence diagrams is not a vector representation of the topological features, it cannot be directly fed into a machine learning algorithm as a feature vector. However, the K-medoids clustering algorithm can take as input a distance matrix rather than a set of feature vectors. The pairwise bottleneck distance between stacked sets of persistence diagrams corresponding to simulations serves as our input for K-medoids. As shown in Table 4, clustering with the bottleneck distance yields higher accuracy in H_0 but lower in H_1 as compared to order parameters. While we compute the order parameter at every time step in order to provide the "best" possible results, we downsample in time with a step size of 10 for the bottleneck computation due to computational complexity. The clustering

TABLE 4. Comparison of the K-medoids clustering accuracy of Experiment 2 of the Euclidean distance on order parameters, crocker plots, and crocker stacks, as well as the bottleneck distance on the stacked set of persistence diagrams. All topological representations compute homology in dimensions 0 and 1, denoted H_0 and H_1. The top accuracy scores of each column are bolded. The parentheses on the order parameter row indicate that the same computation is performed in both columns since order parameters do not incorporate homology dimensions.

	H_0	H_1
Order Parameters	(0.61)	(0.61)
Crocker Plots	**0.67**	**0.71**
Crocker Stacks	**0.67**	0.63
Stacked Persistence Diagrams	**0.67**	0.49

accuracy for stacked persistence diagrams is either the same as or lower than crocker plots.

We now contrast the computational complexities for computing bottleneck distances of stacked persistence diagrams and Euclidean distances of crocker plots or stacks. Both processes start with the persistent homology data from our simulations over time. While we can directly compute the bottleneck distance between simulations from the interval data at a fixed time and then find the supremum over all times, in the crocker representations, we first need to transform the interval data to crocker plots or stacks, vectorize the crocker representation, and then compute the pairwise Euclidean distance between simulations. Even though there is a transformational step to convert the interval data into the crocker representations, the computational time for bottleneck distance matrices is roughly four orders of magnitude larger than for Euclidean distance matrices, *including* the transformation to crockers. Since clustering with stacked persistence diagrams does not yield higher clustering accuracy for this experiment and is far more time-intensive than the crocker representations, we posit that crockers may serve as a better means for parameter identification. Further, as crockers are vector representations, they are more amenable to a host of machine learning tools.

This concludes our experimental analysis of parameter identification of the Vicsek model using order parameters, crocker plots, crocker stacks, and stacked sets of persistence diagrams.

6. Distances between metric spaces and persistence modules.

We now survey a variety of notions of distances between metric spaces and persistence modules, which will be useful for describing the continuity properties of crocker stacks in Section 7.

6.1. The Hausdorff distance.

In order to introduce the stability of persistent homology, we will need some notion of distance on metric spaces. The Hausdorff distance [46] measures the distance between metric spaces X and Y that are "aligned". More precisely, we mean that X and Y are subsets of a larger metric space (Z, d) that contains both X and Y as submetric spaces. For $X \subseteq Z$ and $\delta > 0$, let $X^\delta := \{z \in Z \mid d(z, x) \leq \delta \text{ for some } x \in X\}$ denote the δ-*offset* of X in Z.

Definition 6.1. If X and Y are two subsets of a metric space Z, then the *Hausdorff distance* between X and Y is $d_H^Z(X,Y) = \inf\{\delta > 0 \mid X \subseteq Y^\delta \text{ and } Y \subseteq X^\delta\}$, which is equivalent to

$$d_H^Z(X,Y) = \max\left\{ \sup_{x\in X} \inf_{y\in Y} d(x,y),\ \sup_{y\in Y} \inf_{x\in X} d(x,y) \right\}.$$

An illustration of the Hausdorff distance is shown in Figure 15.

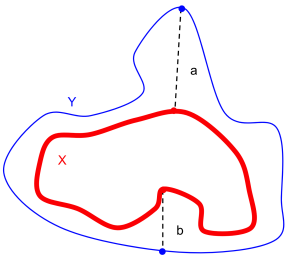


FIGURE 15. Let $Z = \mathbb{R}^2$ with the Euclidean metric. The thicker solid curve (in red) represents subset X of Z, and the thinner solid curve (in blue) represents subset Y of Z. To compute the Hausdorff distance between X and Y, we first take the supremum over all points in Y of the distance to the closest point in X. In this figure, the distance is $a = \sup_{y\in Y} \inf_{x\in X} d(x,y)$. Then, we do the same for the supremum over all points in X of the distance to the closest point in Y, as shown by $b = \sup_{x\in X} \inf_{y\in Y} d(x,y)$. Finally, we take the maximum of the two suprema, $d_H^Z(X,Y) = \max\{a,b\} = a$.

The Hausdorff distance is an extended pseudo-metric on the subsets of Z; two sets have Hausdorff distance zero if and only if they have the same closure, and the Hausdorff distance between unbounded sets can be infinite. When restricted to the set of all non-empty compact subsets of Z, the Hausdorff distance is in fact a metric. We often write d_H, instead of d_H^Z, in order to simplify notation when space Z is clear.

6.2. **Gromov–Hausdorff distance.** The Gromov–Hausdorff distance [8] allows us to define a notion of distance between two metric spaces that are not aligned in any sense.

Definition 6.2. The *Gromov–Hausdorff* distance between two metric spaces X and Y is

$$d_{\mathrm{GH}}(X,Y) = \inf_{Z,f,g} d_H^Z(f(X), g(Y)),$$

where the infimum is taken over all possible metric spaces Z and isometric embeddings $f\colon X \to Z$ and $g\colon Y \to Z$.

The Gromov–Hausdorff distance is an extended pseudo-metric on metric spaces (non-isometric spaces, such as the rationals and the reals, can have Gromov–Hausdorff distance zero). The Gromov–Hausdorff distance is a metric when restricted to the quotient space of compact metric spaces under the equivalence relation of isometry.

6.3. **The stability of persistent homology.** By the stability of persistent homology [17, Theorem 5.2], if X and Y are compact metric spaces, then the bottleneck distance satisfies

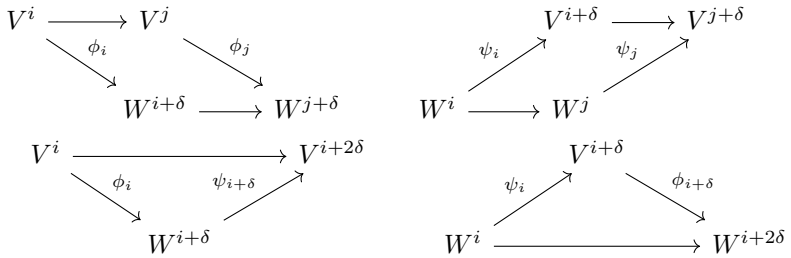
$$d_b(\mathrm{PH}(\mathrm{VR}(X)), \mathrm{PH}(\mathrm{VR}(Y))) \leq 2d_{\mathrm{GH}}(X, Y).$$

An analogous bound is true if Vietoris–Rips complexes are replaced with Čech complexes.

In many applications, spaces X and Y are usually already embedded in the same space, in which one can use the bound $d_{\mathrm{GH}} \leq d_H$ in order to get a lower bound on the Hausdorff distance between these particular embeddings. The stability theorem states that if two metric spaces are close, then the bottleneck distance between their persistence diagrams (using the Vietoris–Rips or Čech complex to construct the filtration) will also be close. Stability is a useful property as it allows for small perturbations of the inputs, and as such, stability of persistence modules has led to their effectiveness in data analysis. One of our motivations for defining crocker stacks is their analogous continuity properties.

6.4. **The interleaving distance.** A closely related notion to the bottleneck distance between persistence diagrams associated to persistence modules (Section 2.4) is that of δ-interleaving [49].

Definition 6.3. For two persistence modules V and W, a *δ-interleaving* (for $\delta \geq 0$) is given by two families of linear maps $(\phi_i : V^i \to W^{i+\delta})$ and $(\psi_i : W^i \to V^{i+\delta})$ such that the following diagrams commute for all $i \leq j$:



Note that a 0-interleaving between two persistence modules is nothing more than an isomorphism between them. The *interleaving distance* between persistence modules V and W is defined as

$$d_I(V, W) = \inf\{\delta \geq 0 \mid \text{there is a } \delta\text{-interleaving between } V \text{ and } W\}.$$

Roughly speaking, one should think of the interleaving distance between two persistence modules as a measure of how far they are from being isomorphic.

In this paper, we do not directly use the interleaving distance on persistence modules. However, by the isometry theorem of Lesnick [38], the interleaving distance is equal to the bottleneck distance, namely $d_I = d_b$. As such, whenever we invoke the bottleneck distance, we could alternatively invoke the interleaving distance on persistence modules. The morphisms induced on the persistence modules by interleaving will allow us to prove Lemma 6.4, which is helpful for describing the continuity of crocker stacks.

6.5. **The rank invariant.** Let V be a persistence module. We recall that the collection of all natural numbers $\mathrm{rank}(V(\varepsilon) \to V(\varepsilon'))$ for all choices of $\varepsilon \leq \varepsilon'$ is called the *rank invariant*. The rank invariant is equivalent to the peristence barcode, in the sense that it is possible to obtain either one from the other [14,

Theorem 12]. An interesting historical comment is that persistent homology of a finite set of points X was first defined as the collection of ranks of all maps of the form $H(\mathrm{VR}(X;\varepsilon-\alpha)) \to H(\mathrm{VR}(X;\varepsilon+\alpha))$, as opposed to a persistence module, barcode, or diagram. See for example the definition on page 151 of [26], where their i is $\varepsilon-\alpha$, and where their j is $\varepsilon+\alpha$.

Given a persistence module V, we encode the rank invariant for V as a function $g_V\colon [0,\infty)\times[0,\infty)\to\mathbb{N}$, where $g_V(\varepsilon,\alpha)=\mathrm{rank}(V(\varepsilon-\alpha)\to V(\varepsilon+\alpha))$. The function g_V is a function on a 2D domain, where the two dimensions are scale ε and persistence parameter α.

We describe a connection between one time-slice of a crocker stack and the rank invariant. Let $\mathbf{V}$ be a time-varying persistence module, and consider a fixed time t. The 2D representation $g_{V_t}\colon [0,\infty)\times[0,\infty)\to\mathbb{N}$, for a fixed time t, is a cross-sectional slice of the 3D crocker stack. Therefore, a crocker stack encodes the rank invariant, and therefore the persistence barcode, of the persistence module V_t at each time t. Persistence lanscapes can be interpreted as a sequence of rank functions, and therefore persistence landscapes [7] are also closely related to these cross-sectional slices (fixing time) in a crocker stack.

6.6. **Relationship of rank invariant and bottleneck distance.** We now describe a relationship between the rank invariant and the bottleneck distance. We consider two persistence modules V and W which decompose into a finite number of intervals, equipped with rank invariants g_V and g_W.

Lemma 6.4. *If $d_b(V,W)\le\delta$, then for all ε and α we have*

- $g_V(\varepsilon,\alpha+\delta)\le g_W(\varepsilon,\alpha)$, *and*
- $g_W(\varepsilon,\alpha+\delta)\le g_V(\varepsilon,\alpha)$.

Proof. By the equivalence between the bottleneck and interleaving distances of persistence modules [38, 49], since $d_b(V,W)\le\delta$, there exist morphisms $\phi_\varepsilon\colon V(\varepsilon)\to W(\varepsilon+\delta)$ and $\psi_\varepsilon\colon W(\varepsilon)\to V(\varepsilon+\delta)$ for all ε, along with the following commutative diagrams.

$$\begin{array}{ccccc}
V(\varepsilon-\alpha-\delta) & & \longrightarrow & & V(\varepsilon+\alpha+\delta) \\
& \searrow & & \nearrow & \\
& & W(\varepsilon-\alpha)\longrightarrow W(\varepsilon+\alpha) & & \\
W(\varepsilon-\alpha-\delta) & & \longrightarrow & & W(\varepsilon+\alpha+\delta) \\
& \searrow & & \nearrow & \\
& & V(\varepsilon-\alpha)\longrightarrow V(\varepsilon+\alpha) & &
\end{array}$$

Indeed, note that the trapezoids above can be obtained by gluing together a parallelogram and a triangle from Definition 6.3. Define V_i^j to be the map from $V(i)$ to $V(j)$. By the first commutative diagram, $V_{\varepsilon-\alpha-\delta}^{\varepsilon+\alpha+\delta}=\psi_{\varepsilon+\alpha}\circ W_{\varepsilon-\alpha}^{\varepsilon+\alpha}\circ\phi_{\varepsilon-\alpha-\delta}$. Since the rank of a composition of linear transformations is at most the minimum rank of the transformations, we have

$$\mathrm{rank}(V_{\varepsilon-\alpha-\delta}^{\varepsilon+\alpha+\delta})\le\min\{\mathrm{rank}(\psi_{\varepsilon+\alpha}),\mathrm{rank}(W_{\varepsilon-\alpha}^{\varepsilon+\alpha}),\mathrm{rank}(\phi_{\varepsilon-\alpha-\delta})\}.$$

Thus, $\mathrm{rank}(V_{\varepsilon-\alpha-\delta}^{\varepsilon+\alpha+\delta})\le\mathrm{rank}(W_{\varepsilon-\alpha}^{\varepsilon+\alpha})$, and so $g_V(\varepsilon,\alpha+\delta)\le g_W(\varepsilon,\alpha)$.

A similar argument for the second diagram shows $g_W(\varepsilon,\alpha+\delta)\le g_V(\varepsilon,\alpha)$. □

We remark that the infimum $\delta \geq 0$ such that $g_V(\varepsilon, \alpha + \delta) \leq g_W(\varepsilon, \alpha)$ and $g_W(\varepsilon, \alpha+\delta) \leq g_V(\varepsilon, \alpha)$ for all ε and α is the *erosion distance* [51, 27, 24, 52] between the two persistence modules V and W. Lemma 6.4 shows that if two persistence modules are close in the bottleneck distance ($\leq \delta$), then their rank invariants are also close in a sense encapsulated by the erosion distance ($\leq \delta$). Could a bound in the reverse direction also be true?

Question 1. *Is the bottleneck distance $d_b(V, W)$ equal to the erosion distance between V and W,* i.e., *the infimum over all $\delta \geq 0$ such that*

- $g_V(\varepsilon, \alpha + \delta) \leq g_W(\varepsilon, \alpha)$, *and*
- $g_W(\varepsilon, \alpha + \delta) \leq g_V(\varepsilon, \alpha)$

for all ε and α?

The answer is no, as shown by an example proposed by Amit Patel and Brittany Terese Fasy.

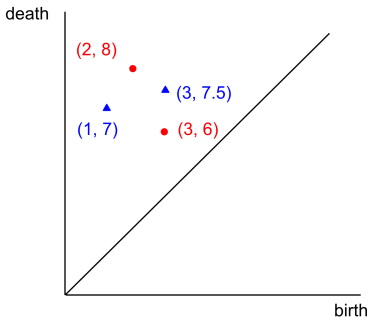


FIGURE 16. Consider the persistence diagrams $\mathrm{Dgm}_k(V) = \{(3, 6), (2, 8)\}$, denoted with red circles, and $\mathrm{Dgm}_k(W) = \{(1, 7), (3, 7.5)\}$, denoted with blue triangles. The bottleneck distance between the two persistence diagrams is 1.5, while the erosion distance is 1.

Example 1. Consider persistence diagrams

$$\mathrm{Dgm}_k(V) = \{(3, 6), (2, 8)\} \text{ and } \mathrm{Dgm}_k(W) = \{(1, 7), (3, 7.5)\},$$

as shown in Figure 16. The bottleneck distance between the two persistence diagrams is 1.5. There are two natural bijections between the two persistence modules, not including matching points with the diagonal (which in this example leads to higher costs). In bijection (i), we match $(3, 6)$ in $\mathrm{Dgm}_k(V)$ with $(3, 7.5)$ in $\mathrm{Dgm}_k(W)$ and $(2, 8)$ in $\mathrm{Dgm}_k(V)$ with $(1, 7)$ in $\mathrm{Dgm}_k(W)$. The L_∞ distance between matched points is 1.5. In bijection (ii), we match $(3, 6)$ in $\mathrm{Dgm}_k(V)$ with $(1, 7)$ in $\mathrm{Dgm}_k(W)$ and $(2, 8)$ in $\mathrm{Dgm}_k(V)$ with $(3, 7.5)$ in $\mathrm{Dgm}_k(W)$. The L_∞ distance between matched points is 2. The bottleneck distance is the infimum L_∞ distance between matched points over all bijections, which is 1.5.

The erosion distance is the infimum $\delta \geq 0$ such that for all ε and α, we have $g_V(\varepsilon, \alpha + \delta) \leq g_W(\varepsilon, \alpha)$, and $g_W(\varepsilon, \alpha + \delta) \leq g_V(\varepsilon, \alpha)$. For this example, $\delta = 1$, for the following reasons: The maximum interval over which V has rank two is $[3, 6]$, whereas the maximum interval over which W has rank two is $[3, 7]$, and these regions differ by at most $\delta = 1$ in their endpoints. Similarly, the maximum interval

over which V has rank at least one is over $[2, 8]$, whereas the maximal intervals over which W has rank at least one is over either $[1, 7]$ or $[3, 7.5]$: enlarging $[2, 8]$ by $\delta = 1$ on either endpoint covers either of these intervals in W, and enlarging $[1, 7]$ by $\delta = 1$ on either endpoint covers $[2, 8]$. This is an intuitive explanation why for all ε and α, we have

$$g_V(\varepsilon, \alpha + 1) = \operatorname{rank}(V^{\varepsilon+\alpha+1}_{\varepsilon-\alpha-1}) \leq \operatorname{rank}(W^{\varepsilon+\alpha}_{\varepsilon-\alpha}) = g_W(\varepsilon, \alpha) \quad \text{and}$$
$$g_W(\varepsilon, \alpha + 1) = rank(W^{\varepsilon+\alpha+1}_{\varepsilon-\alpha-1}) \leq \operatorname{rank}(V^{\varepsilon+\alpha}_{\varepsilon-\alpha}) = g_V(\varepsilon, \alpha).$$

In this example, the bottleneck distance (1.5) is larger than the erosion distance (1), answering Question 1 in the negative.

According to personal correspondence with Patel and Fasy, there is an $O(n \log n)$ algorithm for computing the erosion distance, where n is the number of points in the diagram, which is much faster than computing the bottleneck distance. While Question 1 reveals that the two distances are not equivalent, the erosion distance could serve as a bound on the bottleneck distance. The rank invariants are also Möbius inversions of persistence diagrams [51, 40].

7. **Continuity of crocker stacks.** If two time-varying metric spaces are close to one another, then it turns out that the resulting crocker stacks are also (in some sense) close to one another. This is referred to as the *continuity* of crocker stacks. We explain why crocker plots are not continuous, before describing the sense in which crocker stacks are continuous.

7.1. **Discontinuity of crocker plots.** We first point out that crocker plots are not continuous.

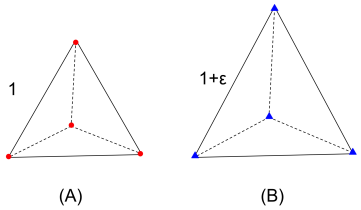


FIGURE 17. Suppose A and B are dynamic metric spaces. The distance between any two of the four round points (in red) in A is 1 for all times t. The distance between any two of the four triangular points (in blue) in B is $1 + \varepsilon$ for all times t. At an arbitrary time step t and scale parameter $1 + \frac{\varepsilon}{2}$, we have $\beta_0^A = 1$ and $\beta_0^B = 4$.

Example 2. Suppose we have two dynamic metric spaces, each with 4 points in the metric space, as shown in Figure 17. In the first dynamic metric space A, the distance between any two points is 1 for all times t. In the second dynamic metric space B, the distance between any two points is $1+\varepsilon$ for all times t. The two dynamic metric spaces are ε-close in the Gromov–Hausdorff distance. However, at scale parameter $1 + \frac{\varepsilon}{2}$, the first dynamic metric space A at any time has 0-dimensional Betti number $\beta_0^A = 1$. Yet at any time in the second dynamic metric space, $\beta_0^B = 4$ at this same scale value. Thus, these Betti numbers differ by $\beta_0^B - \beta_0^A = 4 - 1 = 3$ at scale $1 + \frac{\varepsilon}{2}$. By increasing $n = 4$ points to (say) $n = 1,000,000$ points or beyond,

we can make the values of these Betti numbers as far apart as we want. Under most notions of matrix distance, this would make the distance between the crocker plot matrices as far apart as we want, all while keeping the metric spaces within ε in the Gromov–Hausdorff distance. This example does not rely on time; it is really an example showing why Betti curves are not stable (in the traditional sense of L_∞ distance between curves).

7.2. **Distances between time-varying metric spaces.** In order to describe the continuity properties of crocker stacks, we begin with some preliminaries on distances between time-varying metric spaces and time-varying persistence modules.

Definition 7.1. Let $\mathbf{X}$ and $\mathbf{Y}$ be continuous time-varying metric spaces over $t \in [0, T]$, and fix $1 \leq p \leq \infty$. The p-Gromov–Hausdorff distance between $\mathbf{X}$ and $\mathbf{Y}$ is

$$d_{\mathrm{GH}}^p(\mathbf{X}, \mathbf{Y}) = \left(\int_0^T d_{\mathrm{GH}}(X_t, Y_t)^p \, dt \right)^{1/p}.$$

When $p = \infty$ we have $d_{\mathrm{GH}}^\infty(\mathbf{X}, \mathbf{Y}) = \sup_t d_{\mathrm{GH}}(X_t, Y_t)$.

To see that this is well-defined, note that since $\mathbf{X}$ and $\mathbf{Y}$ are continuous, the function $[0, T] \to \mathbb{R}$ defined by $t \mapsto d_{\mathrm{GH}}(X_t, Y_t)^p$ is a continuous function over a closed interval, and hence is integrable.

We remark that this is only a pseudo-metric, not an actual metric. Indeed, as pointed out in Figure 1 of [37], two distinct time-varying metric spaces that are not qualitatively similar can have Gromov–Hausdorff distance zero from each other at each time t. The distances between multiparameter rank functions introduced in [37] are also stable with respect to more refined notions of distance between time-varying metric spaces. In this paper, we restrict attention to the weaker Definition 7.1 and show that crocker stacks, which are amenable for machine learning tasks, are furthermore a continuous topological invariant.

7.3. **Distances between time-varying persistence modules.** We define an L_p bottleneck distance between corresponding time-varying persistence modules.

Definition 7.2. Let $\mathbf{V}$ and $\mathbf{W}$ be continuous time-varying persistence modules over $t \in [0, T]$, and fix $1 \leq p \leq \infty$. The p-bottleneck distance between $\mathbf{V}$ and $\mathbf{W}$ is $d_b^p(\mathbf{V}, \mathbf{W}) = (\int_0^T d_b(V_t, W_t)^p \, dt)^{1/p}$. When $p = \infty$ we have $d_b^\infty(\mathbf{V}, \mathbf{W}) = \sup_t d_b(V_t, W_t)$.

Since $\mathbf{V}$ and $\mathbf{W}$ are continuous, the function $[0, T] \to \mathbb{R}$ defined by $t \mapsto d_b(V_t, W_t)^p$ is a continuous function over a closed interval, and hence is integrable.

Recall that our time-varying metric spaces $\mathbf{X}$ are defined to have the property that X_t is compact for all $t \in [0, T]$.

Lemma 7.3. *If $\mathbf{X}$ and $\mathbf{Y}$ are continuous time-varying metric spaces over $t \in [0, T]$, then for all $1 \leq p \leq \infty$ we have $d_b^p(\mathrm{PH}(\mathrm{VR}(\mathbf{X})), \mathrm{PH}(\mathrm{VR}(\mathbf{Y}))) \leq 2 d_{\mathrm{GH}}^p(\mathbf{X}, \mathbf{Y})$.*

An analogous bound is true if Vietoris–Rips complexes are replaced with Čech complexes.

Proof. Let $p < \infty$. For any $t \in [0, T]$, we have $d_b(\mathrm{PH}(\mathrm{VR}(X_t)), \mathrm{PH}(\mathrm{VR}(Y_t))) \leq 2 d_{\mathrm{GH}}(X_t, Y_t)$ by the stability of persistent homology. Integrating over all $t \in [0, T]$

gives

$$d_b^p(\mathrm{PH}(\mathrm{VR}(\mathbf{X})),\mathrm{PH}(\mathrm{VR}(\mathbf{Y}))) = \left(\int_0^T d_b\Big(\mathrm{PH}(\mathrm{VR}(X_t)),\mathrm{PH}(\mathrm{VR}(Y_t))\Big)^p\, dt\right)^{1/p}$$
$$\leq \left(\int_0^T \Big(2d_{\mathrm{GH}}(X_t,Y_t)\Big)^p\, dt\right)^{1/p} = 2\left(\int_0^T d_{\mathrm{GH}}(X_t,Y_t)^p\, dt\right)^{1/p} = 2d_{\mathrm{GH}}^p(\mathbf{X},\mathbf{Y}).$$

The same proof works for $p=\infty$ by replacing integrals with supremums. Indeed,

$$d_b^\infty(\mathrm{PH}(\mathrm{VR}(\mathbf{X})),\mathrm{PH}(\mathrm{VR}(\mathbf{Y}))) = \sup_t d_b\Big(\mathrm{PH}(\mathrm{VR}(X_t)),\mathrm{PH}(\mathrm{VR}(Y_t))\Big)$$
$$\leq 2\sup_t d_{\mathrm{GH}}(X_t,Y_t) = 2d_{\mathrm{GH}}^\infty(\mathbf{X},\mathbf{Y}).$$

□

Lemma 7.3 gives a notion of continuity for stacked sequences of persistence diagrams. By contrast, the vines in a vineyard are not stable as curves in time, as explained in Section 3.1.

7.4. **Continuity of crocker stacks.** The continuity of a stacked set of persistence diagrams in the section above implies a continuity result for crocker stacks. Recall from Definition 4.1 that if $\mathbf{V}$ is a time-varying persistence module, then

$$f_{\mathbf{V}}(t,\varepsilon,\alpha) := \mathrm{rank}\,(V_t(\varepsilon-\alpha)\to V_t(\varepsilon+\alpha)) = g_{V_t}(\varepsilon,\alpha).$$

Suppose two time-varying metric spaces $\mathbf{X}$ and $\mathbf{Y}$ have the property that X_t and Y_t are within Hausdorff distance 2δ at all times t. The interleaving distance between the persistence modules $\mathrm{PH}(\mathrm{VR}(X_t))$ and $\mathrm{PH}(\mathrm{VR}(Y_t))$ is at most δ at all times t. Hence, for all t, ε, and α, we have that $f_{\mathbf{X}}(t,\varepsilon,\alpha+\delta) \leq f_{\mathbf{Y}}(t,\varepsilon,\alpha)$ and $f_{\mathbf{Y}}(t,\varepsilon,\alpha+\delta) \leq f_{\mathbf{X}}(t,\varepsilon,\alpha)$, where by an abuse of notation we let $f_{\mathbf{X}}$ and $f_{\mathbf{Y}}$ denote $f_{\mathrm{PH}(\mathrm{VR}(\mathbf{X}))}$ and $f_{\mathrm{PH}(\mathrm{VR}(\mathbf{Y}))}$. This can be thought of as a notion of continuity, since it says the crocker stack $f_{\mathbf{X}}$ for $\mathbf{X}$ is in some sense "close" to the crocker stack $f_{\mathbf{Y}}$ for $\mathbf{Y}$. In this subsection, we provide proofs for these observations.

Lemma 7.4. *If $\mathbf{V}$ and $\mathbf{W}$ are time-varying persistence modules, and if $d_b^\infty(\mathbf{V},\mathbf{W}) \leq \delta$, then for all t, ε, and α we have*

- $f_{\mathbf{V}}(t,\varepsilon,\alpha+\delta) \leq f_{\mathbf{W}}(t,\varepsilon,\alpha)$, *and*
- $f_{\mathbf{W}}(t,\varepsilon,\alpha+\delta) \leq f_{\mathbf{V}}(t,\varepsilon,\alpha)$.

Proof. By hypothesis, we have $d_b(V_t,W_t) \leq \delta$ for all $t\in[0,T]$. From Lemma 6.4 we have $g_{V_t}(\varepsilon,\alpha+\delta) \leq g_{W_t}(\varepsilon,\alpha)$ and $g_{W_t}(\varepsilon,\alpha+\delta) \leq g_{V_t}(\varepsilon,\alpha)$ for all $t\in[0,T]$. The conclusion follows since, by definition, we have $f_{\mathbf{V}}(t,\varepsilon,\alpha) = g_{V_t}(\varepsilon,\alpha)$ and $f_{\mathbf{W}}(t,\varepsilon,\alpha) = g_{W_t}(\varepsilon,\alpha)$ for all t, ε, and α. □

Note that a version of the above lemma for d_b^p instead of d_b^∞ would not give inequalities for each t but instead a single pair of inequalities that are integrated (in an L_p sense) over all t.

The following theorem says that if the time-varying metric spaces $\mathbf{X}$ and $\mathbf{Y}$ are nearby, then their crocker stacks are also close. Recall that our time-varying metric spaces are defined to be compact at each point in time.

Theorem 7.5 (Continuity theorem for crocker stacks)**.** *If $\mathbf{X}$ and $\mathbf{Y}$ are time-varying metric spaces, and if $d_{\mathrm{GH}}^\infty(\mathbf{X},\mathbf{Y}) \leq \delta/2$, then the crocker stacks for $\mathbf{X}$ and $\mathbf{Y}$ are close in the sense that for all t, ε, and α, we have*

- $f_{\mathbf{X}}(t, \varepsilon, \alpha + \delta) \leq f_{\mathbf{Y}}(t, \varepsilon, \alpha)$, *and*
- $f_{\mathbf{Y}}(t, \varepsilon, \alpha + \delta) \leq f_{\mathbf{X}}(t, \varepsilon, \alpha)$.

Proof. By Lemma 7.3 we have that $d_b^\infty(\mathrm{PH}(\mathrm{VR}(\mathbf{X})), \mathrm{PH}(\mathrm{VR}(\mathbf{Y}))) \leq \delta$, and hence the conclusion follows from Lemma 7.4 with $\mathbf{V} = \mathrm{PH}(\mathrm{VR}(\mathbf{X}))$ and $\mathbf{W} = \mathrm{PH}(\mathrm{VR}(\mathbf{Y}))$. □

An analogous result is true if crocker stacks are defined using Čech complexes in place of Vietoris–Rips complexes.

Though crocker stacks are continous in the above sense, we want to acknowledge that they are not continuous when they are interpreted as vectors equipped with the Euclidean norm, which was the norm we used on crocker stacks in Section 5. More work remains to be done on effectively and stably vectorizing time-varying topological summaries for use in machine learning applications.

8. **Conclusion.**

In this paper, we have provided an overview of topological tools for summarizing time-varying metric spaces, developed a new tool called the crocker stack, investigated the discriminative power of topological descriptors on a parameter recovery task, and discussed notions of continuity for these representations.

The crocker plot introduced in [55] is a topological summary of time-varying data. It has been shown in [55, 56, 6] to be useful in exploratory data analysis, statistical tests, and machine learning tasks. However, the crocker plot is not stable in any sense. We proposed the crocker stack as an alternative, which—like the crocker plot—can be discretized and treated as a vector in Euclidean space, making it useful in machine learning. Yet, the crocker stack also satisfies a continuity property, albeit with respect to a non-Euclidean metric.

A crocker stack is a 3D topological representation of a time-varying metric space that, like the crocker plot, displays all times in a single frame indexed by smoothing parameter α. This may be better for visualization purposes than, say, the vineyard representation. Vineyards or stacked sets of persistence diagrams would have to be further vectorized for use in machine learning, perhaps by using say, persistence images [2]. The stable multiparameter rank function and distance measures in [37] are more discriminatory for time-varying metric spaces than the (pseudo-)distances we consider, but it is not obvious how to vectorize them for use in machine learning.

Through computational experiments, we show that crocker plots and stacks are more effective than the alignment order parameter, a traditional method derived from physics, at parameter identification in an ubiquitous model of biological aggregations. However, computing crocker plots and stacks is more time-intensive than computing order parameters. At each time step, a single number—the normalized average velocity—is computed in the order parameter, while persistent homology is computed in order to produce crocker plots and stacks. In our experiments, computing persistent homology is roughly two orders of magnitude more expensive than computing order parameters. The discriminative capability of the crocker representations may provide benefits that outweigh this computational complexity, however.

We end with a collection of questions and possible directions for researchers to explore.

1. How do different choices of metrics affect the discriminatory power of crocker stacks in Section 5? For every $1 \leq p \leq \infty$ and $1 \leq q \leq \infty$, there exists a (p, q) metric on crocker stacks: to compare two fixed persistence diagrams, we can

choose any $1 \leq p \leq \infty$ and use an L_p Wasserstein distance. To compare two time-varying persistence diagrams, we then could weight these distances over all times by choosing any $1 \leq q \leq \infty$ averaging.

2. What is the discriminatory power of a time-varying erosion distance say, for example, in the experiments in Section 5?
3. How well do time-varying persistence images [2] or time-varying persistence landscapes [7] work for machine learning classification tasks?
4. What are other notions of time-varying topological invariants that are both stable and also vectorizable in a way that is useful in machine learning?

Acknowledgments. We are grateful to Matraiyee Deka, Brittany Terese Fasy, Tom Halverson, Michael Lesnik, Dmitriy Morozov, and Amit Patel for helpful conversations.

REFERENCES

[1] H. Adams and G. Carlsson, Evasion paths in mobile sensor networks, *International Journal of Robotics Research*, **34** (2015), 90–104.

[2] H. Adams, T. Emerson, M. Kirby, R. Neville, C. Peterson, P. Shipman, S. Chepushtanova, E. Hanson, F. Motta and L. Ziegelmeier, Persistence images: A stable vector representation of persistent homology, *J. Mach. Learn. Res.*, **18** (2017), Paper No. 8, 35 pp. http://jmlr.org/papers/v18/16-337.html.

[3] H. Adams, D. Ghosh, C. Mask, W. Ott and K. Williams, Efficient evader detection in mobile sensor networks, arXiv preprint, arXiv:2101.09813.

[4] P. Arora, D. Deepali and S. Varshney, Analysis of K-means and K-medoids algorithm for big data, *Procedia Computer Science*, **78** (2016), 507–512.

[5] A. Banman and L. Ziegelmeier, Mind the gap: A study in global development through persistent homology, in *Research in Computational Topology*, Springer, 2018, 125–144.

[6] D. Bhaskar, A. Manhart, J. Milzman, J. T. Nardini, K. M. Storey, C. M. Topaz and L. Ziegelmeier, Analyzing collective motion with machine learning and topology, *Chaos: An Interdisciplinary Journal of Nonlinear Science*, **29** (2019), 123125, 12 pp.

[7] P. Bubenik, Statistical topological data analysis using persistence landscapes, *J. Mach. Learn. Res.*, **16** (2015), 77–102.

[8] D. Burago, Y. Burago and S. Ivanov, *A course in Metric Geometry*, vol. 33, American Mathematical Society, Providence, 2001.

[9] G. Carlsson, Topology and data, *Bull. Amer. Math. Soc. (N.S.)*, **46** (2009), 255–308.

[10] G. Carlsson and V. de Silva, Zigzag persistence, *Found. Comput. Math.*, **10** (2010), 367–405.

[11] G. Carlsson, V. de Silva, S. Kališnik and D. Morozov, Parametrized homology via zigzag persistence, *Algebr. Geom. Topol.*, **19** (2019), 657–700.

[12] G. Carlsson, V. de Silva and D. Morozov, Zigzag persistent homology and real-valued functions, in *Proceedings of the Twenty-Fifth Annual Symposium on Computational Geometry*, ACM, 2009, 247–256.

[13] G. Carlsson, G. Singh and A. Zomorodian, Computing multidimensional persistence, *Algorithms and computation*, 730–739, Lecture Notes in Comput. Sci., 5878, Springer, Berlin, 2009.

[14] G. Carlsson and A. Zomorodian, The theory of multidimensional persistence, *Discrete Comput. Geom.*, **42** (2009), 71–93.

[15] A. Cerri, B. D. Fabio, M. Ferri, P. Frosini and C. Landi, Betti numbers in multidimensional persistent homology are stable functions, *Math. Methods Appl. Sci.*, **36** (2013), 1543–1557.

[16] W. Chachólski, M. Scolamiero and F. Vaccarino, Combinatorial presentation of multidimensional persistent homology, *J. Pure Appl. Algebra*, **221** (2017), 1055–1075.

[17] F. Chazal, V. de Silva and S. Oudot, Persistence stability for geometric complexes, *Geometriae Dedicata*, **174** (2014), 193–214.

[18] D. Cohen-Steiner, H. Edelsbrunner and J. Harer, Stability of persistence diagrams, *Discrete Comput. Geom.*, **37** (2007), 103–120.

[19] D. Cohen-Steiner, H. Edelsbrunner and D. Morozov, Vines and vineyards by updating persistence in linear time, in *Computational Geometry (SCG'06)*, ACM, 2006, 119–126.

[20] P. Corcoran and C. B. Jones, Modelling topological features of swarm behaviour in space and time with persistence landscapes, *IEEE Access*, **5** (2017), 18534–18544.
[21] D. B. Damiano and M. R. McGuirl, A topological analysis of targeted in-111 uptake in SPECT images of murine tumors, *J. Math. Biol.*, **76** (2018), 1559–1587.
[22] V. de Silva and R. Ghrist, Coordinate-free coverage in sensor networks with controlled boundaries via homology, *The International Journal of Robotics Research*, **25** (2006), 1205–1222.
[23] V. de Silva and R. Ghrist, Coverage in sensor networks via persistent homology, *Algebr. Geom. Topol.*, **7** (2007), 339–358.
[24] T. K. Dey and C. Xin, Computing bottleneck distance for 2-d interval decomposable modules, arXiv preprint, arXiv:1803.02869.
[25] M. R. D'Orsogna, Y. L. Chuang, A. L. Bertozzi and L. S. Chayes, Self-propelled particles with soft-core interactions: Patterns, stability, and collapse, *Phys. Rev. Lett.*, **96** (2006), 104302.
[26] H. Edelsbrunner and J. L. Harer, *Computational Topology: An Introduction*, American Mathematical Society, Providence, 2010.
[27] H. Edelsbrunner, D. Morozov and A. Patel, The stability of the apparent contour of an orientable 2-manifold, *Topological Methods in Data Analysis and Visualization. Mathematics and Visualization.*, 27–41, Math. Vis., Springer, Heidelberg, 2011.
[28] B. T. Fasy, J. Kim, F. Lecci, C. Maria, D. L. Millman and V. Rouvreau, Tda: Statistical tools for topological data analysis, https://cran.r-project.org/web/packages/TDA/index.html.
[29] M. Feng and M. A. Porter, Persistent homology of geospatial data: A case study with voting, *SIAM Rev.*, **63** (2021), 67–99.
[30] M. Feng and M. A. Porter, Spatial applications of topological data analysis: Cities, snowflakes, random structures, and spiders spinning under the influence, *Phys. Rev. Research*, **2** (2020), 033426.
[31] R. Ghrist, Barcodes: The persistent topology of data, *ull. Amer. Math. Soc. (N.S.)*, **45** (2008), 61–75.
[32] C. Giusti, L. Papadopoulos, E. T. Owens, K. E. Daniels and D. S. Bassett, Topological and geometric measurements of force-chain structure, *Physical Review E*, **94** (2016), 032909.
[33] I. T. Jolliffe, *Principal Component Analysis*, Springer Verlag, 1986.
[34] T. Kaczynski, K. Mischaikow and M. Mrozek, *Computational Homology*, vol. 157, pringer-Verlag, New York, 2004.
[35] L. Kaufman and P. Rousseeuw, *Clustering by Means of Medoids*, North-Holland, 1987.
[36] W. Kim and F. Mémoli, Stable signatures for dynamic metric spaces via zigzag persistent homology, arXiv preprint, arXiv:1712.04064.
[37] W. Kim and F. Mémoli, Spatiotemporal persistent homology for dynamic metric spaces, *Discrete Comput. Geom.*, **66** (2021), 831–875.
[38] M. Lesnick, The theory of the interleaving distance on multidimensional persistence modules, *Found. Comput. Math.*, **15** (2015), 613–650.
[39] M. Maechler, Finding groups in data: Cluster analysis extended rousseeuw et al, https://cran.r-project.org/web/packages/cluster/cluster.pdf.
[40] A. McCleary and A. Patel, Bottleneck stability for generalized persistence diagrams, *Proc. Amer. Math. Soc.*, **148** (2020), 3149–3161.
[41] A. McCleary and A. Patel, Edit distance and persistence diagrams over lattices, arXiv preprint, arXiv:2010.07337.
[42] E. Miller, Data structures for real multiparameter persistence modules, arXiv preprint, arXiv:1709.08155.
[43] N. Milosavljević, D. Morozov and P. Škraba, Zigzag persistent homology in matrix multiplication time, in *Computational geometry (SCG'11)*, 2011, 216–225.
[44] D. Morozov, Personal communication.
[45] D. Morozov, Dionysus, http://www.mrzv.org/software/dionysus/.
[46] J. R. Munkres, *Topology*, Prentice-Hall Englewood Cliffs, NJ, 1975.
[47] C. Nilsen, J. Paige, O. Warner, B. Mayhew, R. Sutley, M. Lam, A. J. Bernoff and C. M. Topaz, Social aggregation in pea aphids: Experiment and random walk modeling, *PLoS ONE*, **8** (2013), e83343.
[48] N. Otter, M. A. Porter, U. Tillmann, P. Grindrod and H. A. Harrington, A roadmap for the computation of persistent homology, *EPJ Data Science*, **6** (2017), 17.
[49] S. Y. Oudot, *Persistence Theory: From Quiver Representations to Data Analysis*, vol. 209, American Mathematical Society Providence, RI, 2015.

[50] H.-S. Park and C.-H. Jun, A simple and fast algorithm for k-medoids clustering, *Expert Systems with Applications*, **36** (2009), 3336–3341.
[51] A. Patel, Generalized persistence diagrams, *J. Appl. Comput. Topol.*, **1** (2018), 397–419.
[52] V. Puuska, Erosion distance for generalized persistence modules, *Homology Homotopy Appl.*, **22** (2020), 233–254.
[53] M. Scolamiero, W. Chachólski, A. Lundman, R. Ramanujam and S. Öberg, Multidimensional persistence and noise, *Found. Comput. Math.*, **17** (2017), 1367–1406.
[54] B. J. Stolz, H. A. Harrington and M. A. Porter, Persistent homology of time-dependent functional networks constructed from coupled time series, *Chaos*, **27** (2017), 047410, 17 pp.
[55] C. M. Topaz, L. Ziegelmeier and T. Halverson, Topological data analysis of biological aggregation models, *PloS One*, **10** (2015), e0126383.
[56] M. Ulmer, L. Ziegelmeier and C. M. Topaz, A topological approach to selecting models of biological experiments, *PloS One*, **14** (2019), e0213679.
[57] T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen and O. Shochet, Novel type of phase transition in a system of self-driven particles, *Phys. Rev. Lett.*, **75** (1995), 1226–1229.
[58] T. Vicsek and A. Zafeiris, Collective motion, *Physics Reports*, **517** (2012), 71–140.
[59] X. Zhu, Persistent homology: An introduction and a new text representation for natural language processing, in *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.
[60] A. Zomorodian and G. Carlsson, Computing persistent homology, *Discrete Comput. Geom.*, **33** (2005), 249–274.

Received June 2021; revised October 2021; early access December 2021.

E-mail address: xianl@umich.edu
E-mail address: henry.adams@colostate.edu
E-mail address: cmt6@williams.edu
E-mail address: lziegel1@macalester.edu